НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ ІНФОРМАТИКИ ТА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ
(повна назва інституту/факультету)

КАФЕДРА АВТОМАТИКИ ТА УПРАВЛІННЯ В ТЕХНІЧНИХ СИСТЕМАХ
(повна назва кафедри)

«На правах рукопису»
УДК _____

«До захисту допущено»
Завідувач кафедри
_____ Ролік О.І.
(підпис) (ініціали, прізвище)
“___”_____2020 р.

Магістерська дисертація

зі спеціальності (спеціалізації) 126 – Інформаційні системи та технології
(код і назва спеціальності)

на тему: Система керування групою станків з ЧПУ_____
_____

Виконав: студент __2__ курсу, групи ___ІА-92мп_____
(шифр групи)
Косяк Олександр Миколайович _____
(прізвище, ім’я, по батькові) (підпис)

Науковий керівник к.т.н., доцент, Кравець П.І. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Рецензент _____ _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2020 року

Національний технічний університет України
«Київський політехнічний інститут
імені Ігоря Сікорського»

Факультет (інститут)_____Інформатики та обчислювальної техніки___
(повна назва)

Кафедра ____Кафедра автоматики та управління в технічних системах___
(повна назва)

Рівень вищої освіти – другий (магістерський)

Спеціальність __126 – Інформаційні системи та технології_____
(код і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Ролік О.І.
(підпис) (ініціали, прізвище)

«___»_____20__ р.

ЗАВДАННЯ

на магістерську дисертацію студенту

Косяку Олександру Миколайовичу
(прізвище, ім'я, по батькові)

1. Тема дисертації Система керування групою станків з ЧПУ_____
науковий керівник дисертації_Кравець Петро Іванович, к.т.н., доцент,___ ,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «___»_____ 20__ р. №_____

2. Строк подання студентом дисертації _____

3. Об'єкт дослідження автоматизація процесу створення розкладу для групи станків з ЧПУ _____

4. Предмет дослідження (вихідні дані для магістерської дисертації за освітньо-професійною програмою) автоматизована система роботи з групою станків з ЧПУ

5. Перелік завдань, які потрібно розробити Знайти аналоги які на даний час існують на ринку; дослідити існуючі на даний час алгоритми що відповідають за створення розкладів; переглянути існуючі на ринку станки з ЧПУ; дослідити швидкодію роботи системи, та ефективність управління групи станків з ЧПУ

6. Орієнтовний перелік ілюстративного (графічного) матеріалу Схема IDEF0, схема роботи з даними, блок схеми модулів системи, структурна схема, діаграма діяльності, ER та Use-Case діаграми

7. Орієнтовний перелік публікацій _____

_____

8. Консультанти розділів дисертації*

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

9. Дата видачі завдання _____

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Строк виконання етапів магістерської дисертації	Примітка
1	Отримання завдання	20.9.2020	
2	Огляд аналогів	30.10.2020	
3	Розробка системи	10.11.2020	
4	Оформлення документації	01.12.2020	

Студент _____ О.М. Косяк

(підпис) (ініціали, прізвище)

Науковий керівник дисертації _____ П.І. Кравець

(підпис) (ініціали, прізвище)

* Консультантом не може бути зазначено наукового керівника

РЕФЕРАТ

Магістерська дисертація складається з 105 сторінок, 25 зображень, 28 таблиць, посилань на використані джерела та 9 додатків.

Актуальність теми обраної для написання дисертації полягає в створенні системи оптимізації розкладу роботи групи станків з ЧПУ (числовим програмним управлінням) на основі теорії розкладів та ал-горитмів, які використовуються для розв'язання групи задач без переривань. Для створення системи використову-валась мова програмування C#, та принципи ООП (об'єктно-орієнтованого програмування).

Мета роботи - розробка системи керування розкладом роботи групи станків з числовим програмним управлінням (ЧПУ) , що дозволяє мінімізувати час простою станків і підвищити ефективність їх роботи.

Об'єктом дослідження є автоматизовані системи керування станків з ЧПУ.

Предметом дослідження є засоби автоматизації процесу створення розкладу роботи для групи станків з ЧПУ.

Під час розв'язку задач було використано алгоритми теорії розкладів. В загальному випадку розглядають дві множини: перша множина, множина машин (приладів, а в нашому випадку станків з ЧПУ), друга множина, набір робіт (окремих завдань, або пакетів задач, що потрібно виконати для досягнення поставлених цілей, в нашому випаду виготовлення готової деталі з заготовки). Перед нами стоїть задача дискретної оптимізації, побудувати розклад який буде мінімізувати час витрачений на виконання усього списку робіт. Розклад, це набір інструкцій, в яких вказано на яких машинах, і в якому порядку слід виконувати роботи.

Ключові слова: станки з числовим програмним управлінням, оптимізація розкладу, теорія розкладів, алгоритми.

SUMMARY

The master's dissertation consists of 103 pages, 25 images, 28 tables, references to used sources and 9 appendices.

The relevance of the topic chosen for writing the dissertation is to create a system for optimizing the schedule of a group of CNC machines (numerical program control) based on the theory of schedules and algorithms used to solve a group of problems without interruption. The C # programming language and OOP (object-oriented programming) principles were used to create the system.

The purpose of the dissertation is to develop a system for optimizing the work schedule of a group of CNC machines (numerical program control), which will easily increase the efficiency of a group of CNC machines.

The object of research is the automation of the process of creating a schedule for a group of CNC machines.

The subject of the study is an automated system for working with a group of CNC machines.

Schedule theory algorithms were used to solve the problems. In the general case, consider two sets: the first set, a set of machines (devices, and in our case CNC machines), the second set, a set of works (individual tasks, or packages of tasks to be performed to achieve the goals, in our case of manufacturing the finished parts of the workpiece). We are faced with the task of discrete optimization, to build a schedule that will minimize the time spent on the entire list of works. A schedule is a set of instructions that indicate on which machines and in what order the work should be performed.

Keywords: machines with numerical program control, schedule optimization, schedule theory, algorithms.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ І ТЕРМІНІВ

API – application programming interface – інтерфейс взаємодії з програмою.

ЧПУ – числове програмне управління.

БД – база даних.

ПУ – програма управління (станком з ЧПУ).

EF – Entity Framework

IDEF0 – методологія функціонального моделювання

WPF – Windows Presentation Foundation

WF – Windows Forms

UWP – Universal Windows Platform

B&B – алгоритм branch and bound

CNC - computer numerical control

СУБД – система управління базами даних

ARM – архітектура процесорів Advanced RISC Machine

ВСТУП

Станок з ЧПУ (числовим програмним управлінням або CNC – computer numeric control) – складний комплексний технологічний агрегат, що використовується для обробки та виготовлення деталей зі складною геометрією з таких матеріалів як дерево, пластмаса або метал. Сучасні станки з ЧПУ забезпечують високу точність виготовлення деталей. в автоматичному режимі і не потребують безперервного контролю з боку оператора, що забезпечує високу продуктивність їх роботи. З його допомогою можливо створювати деталі, що мають різні форми: від простих геометричних фігур, до тривимірних об'єктів зі складною геометрією.[1]

Однак по закінчені одного технологічного процесу і запуск нового потребує уваги оператора, а при його зайнятості, особливо коли один оператор обслуговує групу станків, може призвести до зайвого простою станків і відповідно зменшення ефективності виробництва. Саме тому система оптимального керування роботою групи станків з ЧПУ, що дозволяє знизити час простою станків при виготовленні партій деталей, є актуальною.

Створення станків з числовим програмним керуванням значно вплинуло не лише на металообробку, але й на роботу з іншими видами матеріалів. Нові покоління станків забезпечують підвищену точність виготовлення деталей, та не потребують безперервного контролю й спостереження оператором, що дозволяє значно підвищити продуктивність роботи. Це в свою чергу знижує вплив «людського фактору», та пов'язані з ним відхилення від планів виробництва. Тож процес виготовлення готових виробів з заготовок відбувається безперервно і в точності слідує заданій програмі, що в свою чергу забезпечує високу точність виробництва.

Саме тому програма, що дозволить знизити час простою станків при виготовленні партій товарів буде актуальна та користуватиметься попитом, адже вона дозволить підвищити ефективність роботи станків з ЧПУ.

Ринок ЧПУ станків являється основою сучасної промисловості, оскільки лише він здатен забезпечити необхідний рівень швидкості й точності обробки деталей, при високих показниках продуктивності та гнучкості виробництва (легко змінювати тип

деталей що виготовляються). Так як технології 3D друку, незважаючи на значний прогрес, все ще залишаються вузькоспеціалізованими і не можуть конкурувати з традиційними обробляючими центрами, особливо в сфері металообробки. За даними інтернет видавництва МНИАП світовий ринок станків з ЧПУ виросте з нинішнього $ 93 млрд до $ 128 млрд до 2026р. Очікується, що в найближче десятиліття основним напрямком розвитку стануть ЧПУ-станки, які використовуються в автомобільній промисловості, зокрема станки що використовуються для формування корпусу автомобіля. Інший перспективний напрямок розвитку, це точне машинобудування. Використовується в авіакосмічній та оборонній промисловості, при виготовленні медичного обладнання та техніки. Складність виробів, використання особливих металів та сплавів, роблять переваги використання станків з ЧПУ вирішальними для широкого їх застосування в наведених вище сегментах ринку.[1]

Предмет дослідження полягає в розробці системи, що на основі введених в неї даних про промислові потужності (кількість станків, їх ефективність) та інформації про групи деталей що потрібно виготовити, зможе автоматично генерувати розклад роботи.

Мета роботи - розробка системи керування розкладом роботи групи станків з числовим програмним управлінням (ЧПУ) , що дозволяє мінімізувати час простою станків і підвищити ефективність їх роботи.

Об’єктом дослідження є автоматизовані системи керування станків з ЧПУ.

Предметом дослідження є засоби автоматизації процесу створення розкладу роботи для групи станків з ЧПУ.

1. АНАЛІЗ ІСНУЮЧИХ РІШЕННЬ

Для роботи станків з ЧПУ потрібні керуючі програми. Вони відповідають за створення макетів майбутніх виробів, введення команд керування станком та правильне розпізнавання інструкцій написаних на спеціальній мові програмування.

Керуючі програми забезпечують автономний, або в деяких випадках напівавтономний процес обробки заготовок. Таке програмне забезпечення включає в себе комплекс команд, що безперервно поступають на станок з ЧПУ.

Ці команди дозволяють в автоматичному режимі виконувати наступні маніпуляції [2]:

1) змінювати положення інструментів в системі координат;
2) змінювати положення деталі в системі координат;
3) контролювати швидкість обробки деталей;
4) змінювати робочий інструмент станка (свердла, фрези), якщо така функція підтримується станком;
5) автоматично передавати готову деталь на конвеєр, якщо така функція підтримується станком;
6) автоматично брати нову заготовку, якщо така функція підтримується станком;

Тож почнемо наш огляд саме з систем керування.

1.1. ArtCAM Pro

ArtCAM Pro — це програмний пакет для просторового моделювання або механічної обробки, котрий дозволяє автоматично генерувати просторові моделі з плоского рисунку і отримувати по ним готові вироби на станках з числовим програмним управлінням. Інтерфейс програми наведено на рисунку 1.1. ArtCAM Pro пропонує користувачу потужний, легкий в використанні набір інструментів

моделювання, що забезпечить дизайнера свободою вибору при створенні складних рельєфів у просторі. [3]

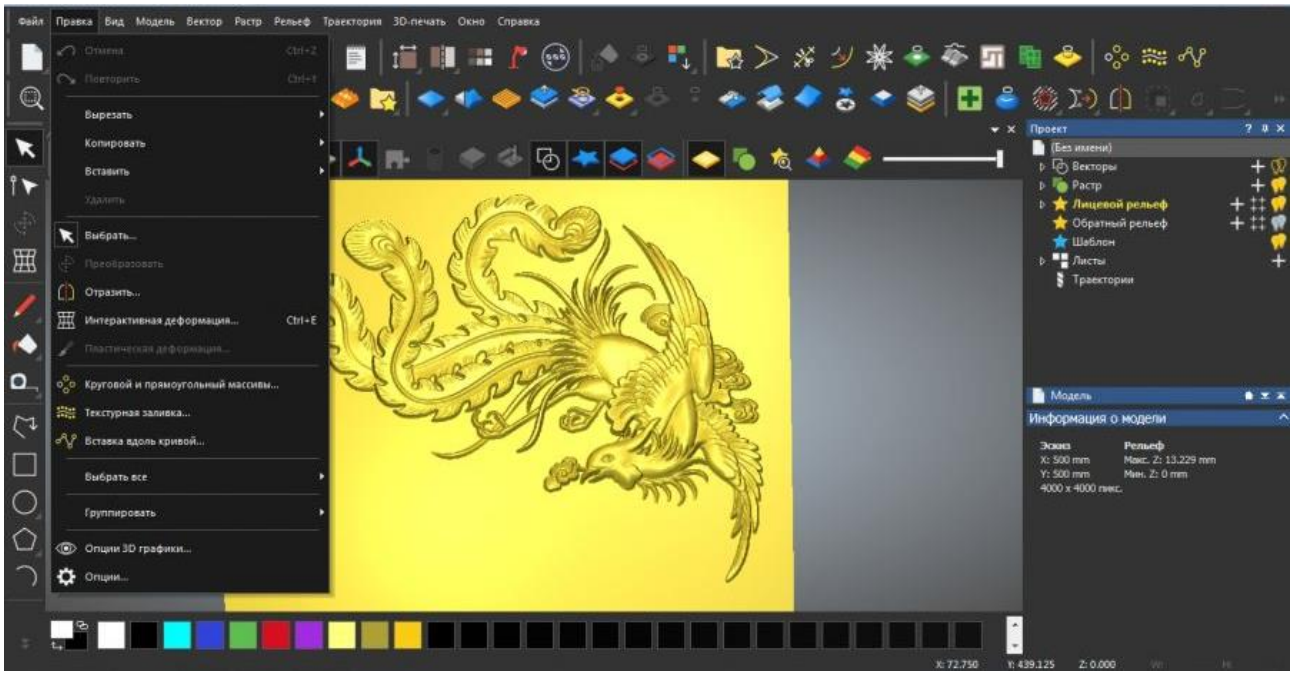

Рисунок 1.1. – Інтерфейс програми ArtCAM Pro 2018

Для створення рельєфу спочатку створюється векторне зображення. Для кожного кольору можливо визначити початкову висоту і тип поверхні. Після чого зображення перетворюється в 3D-рельеф. В процесі роботи можливо змінювати існуючі рельєфи: складати, вирізати, згладжувати або візуально віднімати частини.

Також в програмі присутній примітивний скульптор з невеликим набором доступних функцій.

По створеному 3D-рельефу може бути створена програма управління для станка з ЧПУ.

Переглянемо основні плюси ArtCAM Pro [3]:

- імпорт 2D або 3D векторів, зображень або моделей створених в інших графічних редакторах, підтримуються всі популярні формати;
- наявні інструменти дозволяють створити проект будь-якої складності, від простих до складних;
- бібліотека доступних елементів, що може використовувати кожний користувач під час своєї роботи;

- інструменти що дозволяють знайти помилки, що можуть виникати при імпорті чужих елементів;
- вбудовані системи дозволяють знизити витрати матеріалу під час створення деталей;
- автоматична розбивка траєкторій руху інструментів станка на зони, для полегшення роботи над складними проектами;
- підтримує велику кількість розповсюджених ЧПУ станків;

Тут також варто зазначити, що останній пункт на даний час можна поставити під сумнів, так як компанія AUTODESK, яка й займалася розробкою програми ArtCAM Pro у 2018 році повідомила про завершення підтримки свого програмного продукту. Це і є основним та вирішальним недоліком даної програми, так як вона більше ніким не підтримується, а ринок ЧПУ станків весь час розвивається в майбутньому ця програма втратить свою актуальність, тому зосереджуватись лише на ній немає ніякого сенсу.

1.2. Система LinuxCNC

LinuxCNC керує станками з числовип програмним управлінням. Інтерфейс програми зображено на рисунку 1.2. Він може керувати фрезерними верстатами, токарними верстатами, 3D-принтерами, лазерними різаками, плазмовими різаками, маніпуляторами роботів, гексаподами і так далі.

Основні переваги даної системи [4]:

- працює під Linux (опціонально з розширеннями реального часу);
- проста установка в Debian і Ubuntu або через наші Live / Install DVD / USB-образи;
- приймає введення G-коду, у відповідь запускає верстати з ЧПУ;
- активна спільнота користувачів;
- доступно декілька різних графічних інтерфейсів;
- сумісність з багатьма популярними апаратними інтерфейсами управління верстатом;

- підтримує жорстке нарізування різьблення, компенсацію різця і багато інших розширені функції управління;
- повний вихідний код доступний згідно з умовами GNU GPLv2;

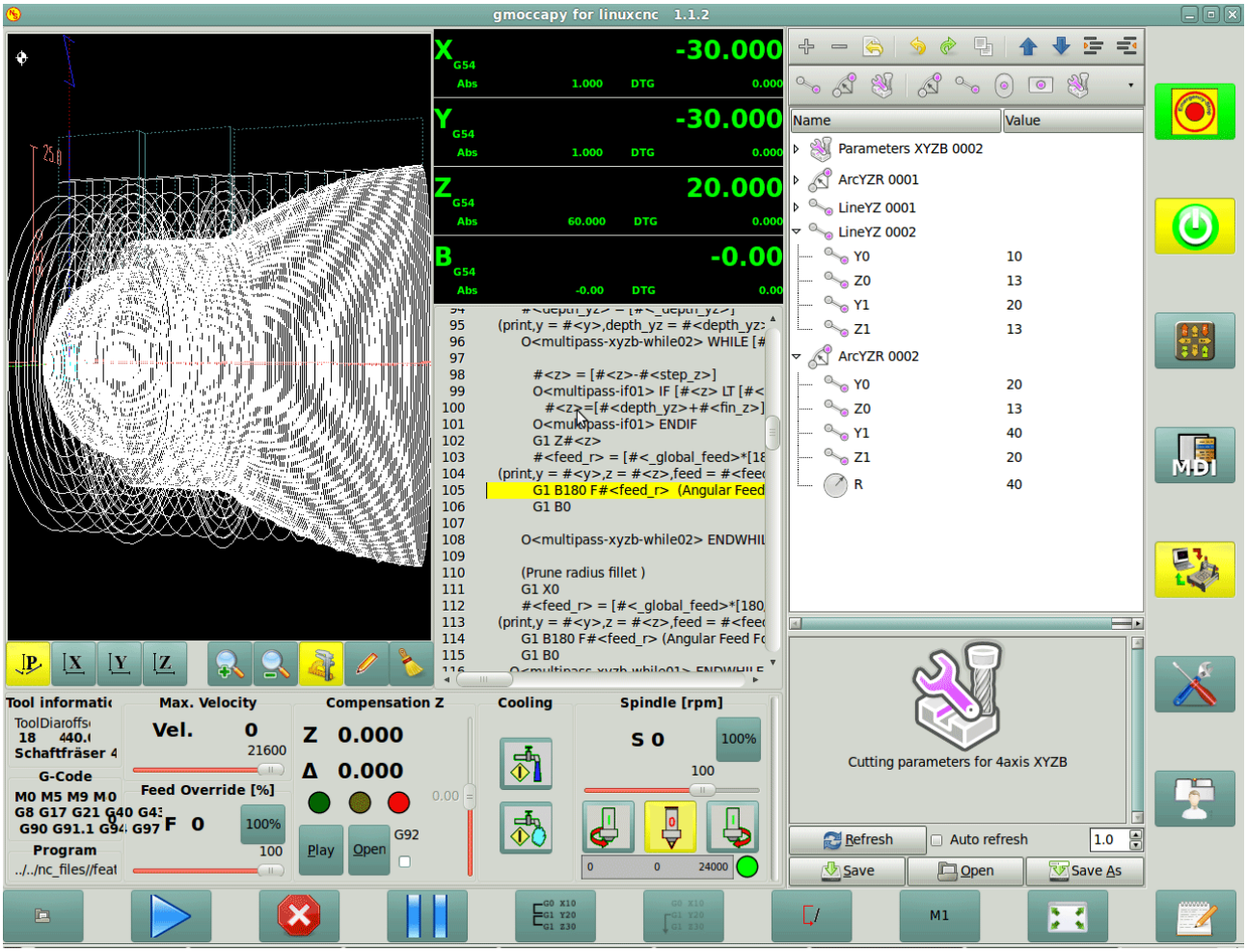

Рисунок 1.2. – Інтерфейс програми LinuxCNC

Як можна побачити, інтерфейс зовсім не інтуїтивно зрозумілий, тому доволі тяжко буде людині яка бачить програму перший раз, як в ній взагалі працювати. Сам інтерфейс програми сильно перевантажений інформацією, і при цьому модель відображається як набір ліній і дуг по яким будуть рухатись інструменти, що може збивати при спробі зрозуміти остаточний вигляд деталі.

Інший очевидний недолік, це те що дана утиліта працює виключно на операційній системі Linux, і не дивлячись на те, що сам по собі цей факт не є

проблемою, для широкого користувача це майже вбиває привабливість даної програми.

1.3. Mach4

Остання версія програмного забезпечення для станків з числовим програмним управлінням. Mach4 абсолютно нове програмне забезпечення в порівнянні з Mach3. Mach4 призначений для роботи з різними типами станків, таких як, фрезерні, плазмові, лазерні, гравірувальні, токарні та інші станки.

В порівнянні з попередньою версією [5]:

- більш точна траєкторія руху інструменту при різних розмірах;
- синхронізований рух шести окремих систем координат, що дозволяє обробляти складні деталі;
- IPC-технологія дозволяє розраховану на багато користувачів і віддалену роботу;
- нова панель інструментів для перегляду, обертання, масштабування траєкторії руху інструмента;
- вдосконалене планування інтерфейсу з допомогою якого можливо змінювати кнопки і індикатори для налаштування налаштування.

У програми є дві версії:

- Mach4 Hobby повнофункціональна версія програмного забезпечення. Може бути корисною для тих, хто зацікавлений у використанні станків з ЧПУ для особистого використання. Підтримка користувачів обмежується здійснюється шляхом електронної пошти та інтернет форумами;
- Mach4 Industrial дорожча версія, яка по суті відрізняється від попередньої в основному об’ємом підтримки що надається користувачу (включаючи підтримку по телефону).
-

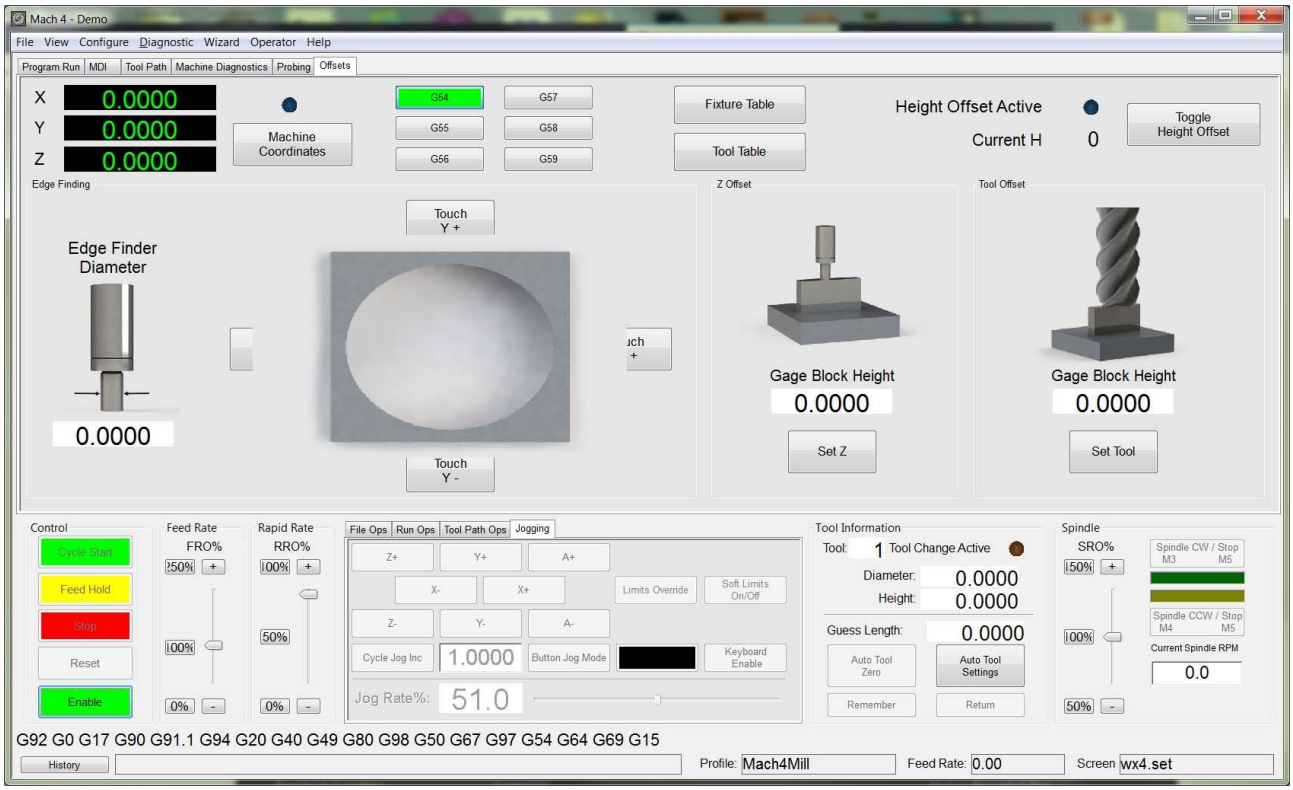

Рисунок 1.3. – Інтерфейс програми Mach4

Проблеми цієї програми схожі з проблемами попереднього аналогу, за тим винятком, що дане програмне забезпечення розраховане на використання в операційних системах сімейства Windows, а не Linux. Це в свою чергу робить дану програму привабливішою для користувачів, але попередня програма розповсюджувалась безкоштовно, а ціна даного додатку, в залежності від версії 200-2000$. Інтерфейс програми зображено на рисунку 1.3. Попередня версія програми, Mach3, значно дешевша, але вона не оновлюється, тож також не може називатись оптимальним рішенням.

1.4. SimplyCam

SimplyCam це повнофункціональна CAD/CAM система для створення 2D і 3D траєкторій обробки на станках з ЧПУ. Побудова геометричних деталей може бути виконана інструментами програми або експортовані з допомогою файлів інших додатків.

Основні характеристики SimplyCam [6]:

- SimplyCam дозволяє відкрвати, створювати, редагувати, а також зберігати креслення в стандартному промисловому форматі – DXF;
- відкриває і працює з файлами SVG;
- також дозволяє конвертувати (векторизувати) растрові зображення (форматів Bmp і Jpeg) в вектори, трасуванням по середній лінії і по контуру, розпізнаючи при цьому дуги і кола;
- наявна можливість маніпулювати векторами (масштабувати, обертати, відзеркалювати, переносити і зсувати), створювати cad-елементи: Лінію, прямокутник, дугу або еліпс, полілінію і сплайн та змінювати їх функціями: скруглити, обрізати, перервати, розширити;
- конвертація будь яких шрифтів True Type в вектори;
- генерує команди G-кодів для фрезерних, гравірувальних, лазерних та плазмових станків, а також токарних станків з ЧПУ;
- включає в себе великий набір відкритих постпроцесорів таких систем управління, як: AUTOGRAV, Emc2, Fadal, Fanuc, Haas, Heidehain, ISO, KCam, Mach2/3, MaxNc, PlasmaCam, ProtoTrak, Selca, Siemens, ShopBot, Tecno-ISEL, TurboCnc та інші;
- можливість налаштовувати інтерфейс під себе, додаючи або видаляючи елементи на панелі швидкого доступу;

Окремо зауважимо що в SimplyCam можливо імпортувати набори команд управління в форматі G-коду із інших систем, візуалізовувати їх виконання (так названа cnc-імітація) і конвертувати їх у геометричний вигляд. Приклад роботи програми зображено на рисунку 1.4.

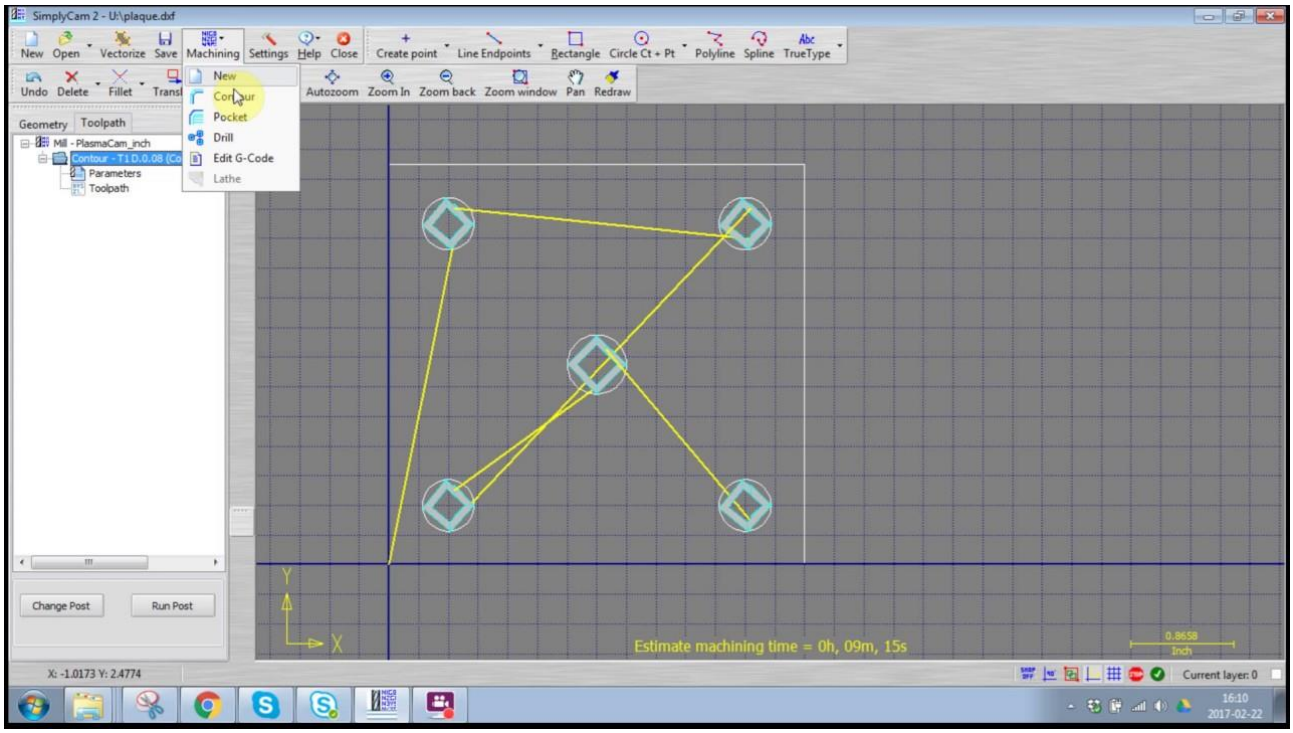

Рисунок 1.4. – Інтерфейс програми SimplyCam

Тобто на основі скрипту, що описує рух інструментів в станку з ЧПУ можемо отримати певну геометричну фігуру, що дозволить побачити деталь, яку отримаємо на виході. При цьому доступно відкриття або зберігання як геометрії так і траєкторій руху, при чому в нас залишається можливість їх переробити в подальшому або перерахувати геометричну модель.

1.5. CutViewer

CutViewer це проста у використанні програма, котра імітує процес обробки з видаленням матеріалу на 2х або 3х вісьному станку з ЧПУ і дає повноцінну твердотільну візуалізацію оброблюваної заготовки і одержуваної деталі. Інтерфейс програми зображено на рисунку 1.5. Використання CutViewer дозволяє збільшити продуктивність праці розробника, заздалегідь розпізнати і усунути помилки програмування, а також скоротити час налагоджувальних робіт на верстаті. Потужні і прості у використанні інструменти програми дозволяють виправляти помилки в траєкторії, виявляти небажані врізання інструменту в заготовку при прискорених

переїздах, проводити розрахунок часу, необхідного на виконання КП. CutViewer може бути налаштований на використання з широким спектром верстатного обладнання [7].

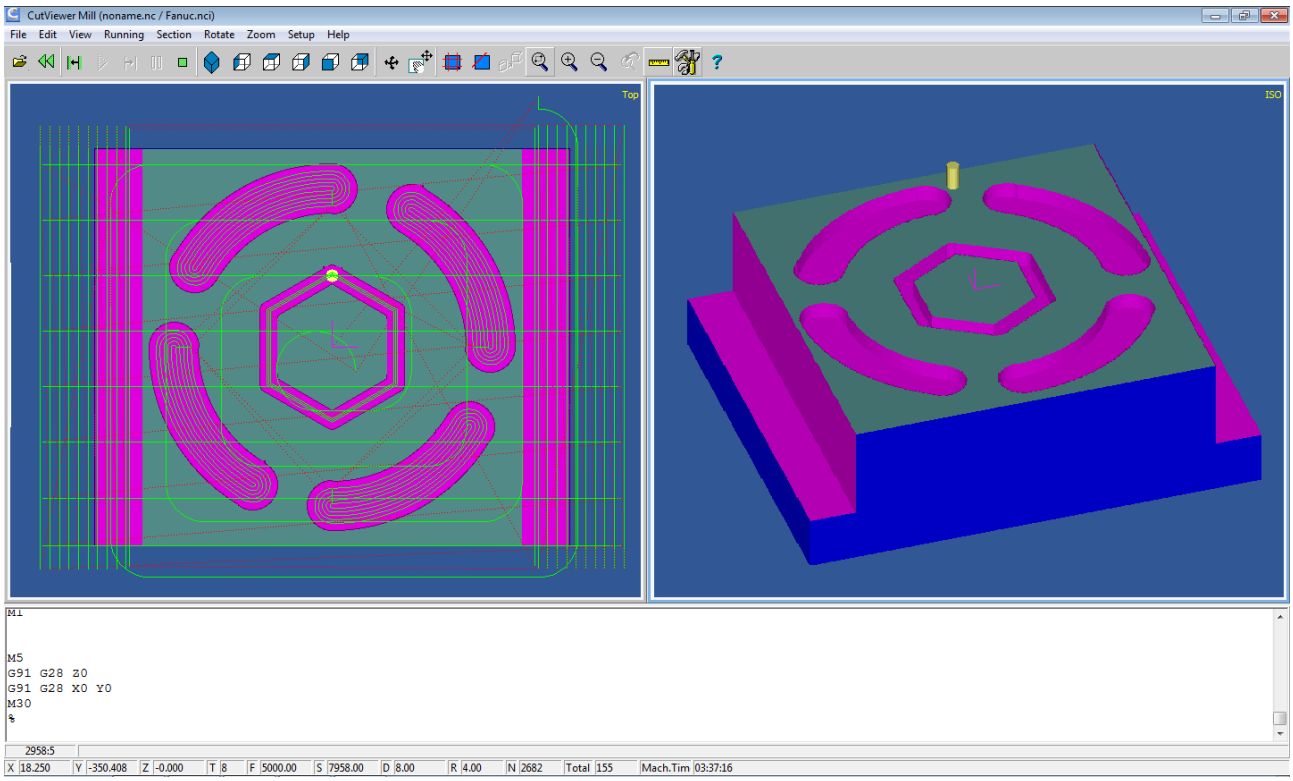

Рисунок 1.5. – Інтерфейс програми SimplyCam

Найважливіша якість даного програмного забезпечення полягає в його здатності надавати допомогу у виявленні помилок і їх усунення до початку обробки дорогого матеріалу, тим самим, заощаджуючи час і гроші виробнику. Потужні можливості редагування забезпечують перегляд і одночасний аналіз і коригування окремих ділянок керуючої програми, що значно спрощує тяжку задачу знаходження проблемних місць в файлі і їх виправлення.

CutViewer дозволяє користувачеві редагувати файли КП, не виходячи з режиму імітації. Це дозволяє проводити корекцію і верифікацію КП оперативно в режимі реального часу. Ключові функції CutViewer - це швидкий перегляд з можливістю тимчасового зупинення (паузи), перегляд обробки з різних точок, збільшення / зменшення зображення, розсічення заготовки на секції для зручності перегляду складних конфігурацій, зміна швидкості імітації обробки і т.д.

CutViewer читає широкий спектр керуючих програм і має дуже просту конфігурацію, яка дозволяє додавати без проблем нові постпроцесори в список сумісних.

1.6. CadStd

CadStd - це проста в освоєнні програма САПР для створення професійних якісних механічних конструкцій, планів будинків, креслень, схем та схем із використанням стандартів креслення ANSI. Полегшена версія безкоштовна і може читати будь-який малюнок, створений версією Pro. CadStd Lite може експортувати файли у форматі DXF, щоб ви могли ділитися своїми малюнками з друзями, які мають інші програми САПР, такі як Autocad. Версія Pro може створювати ізометричні проекції з різних виглядів і має потужні команди, такі як зміщення, обрізка та фаска. Можливість експортувати малюнки у такі формати, як DXF, SVG, HPGL, або копіювати з буферу обміну, щоб вставити зображення в Office та інші програми. Імпортуються файли DXF, HPGL та Gedcom [8].

Одна з головних переваг програми CadStd це можливість дуже швидкого, та простого формування, різних моделей деталей, що може значно прискорити виконання роботи, при цьому не знижуючи деталізацію. По своїй суті це одна з най інтуїтивно зрозуміліших програм для моделювання деталей або систем. Також на офіційному сайті додатку легко знайти всю інформацію по роботі з програмою, що допоможе в освоєнні програми.

Інтерфейс програми CadStd зображено на рисунку 1.6. Основний недолік даної програми полягає в тому, що вона розрахована в основному для роботи з 2D зображеннями, і тривимірні фігури в ній створити неможливо, відповідно неможливо і якось обробляти уже готові тривимірні моделі.

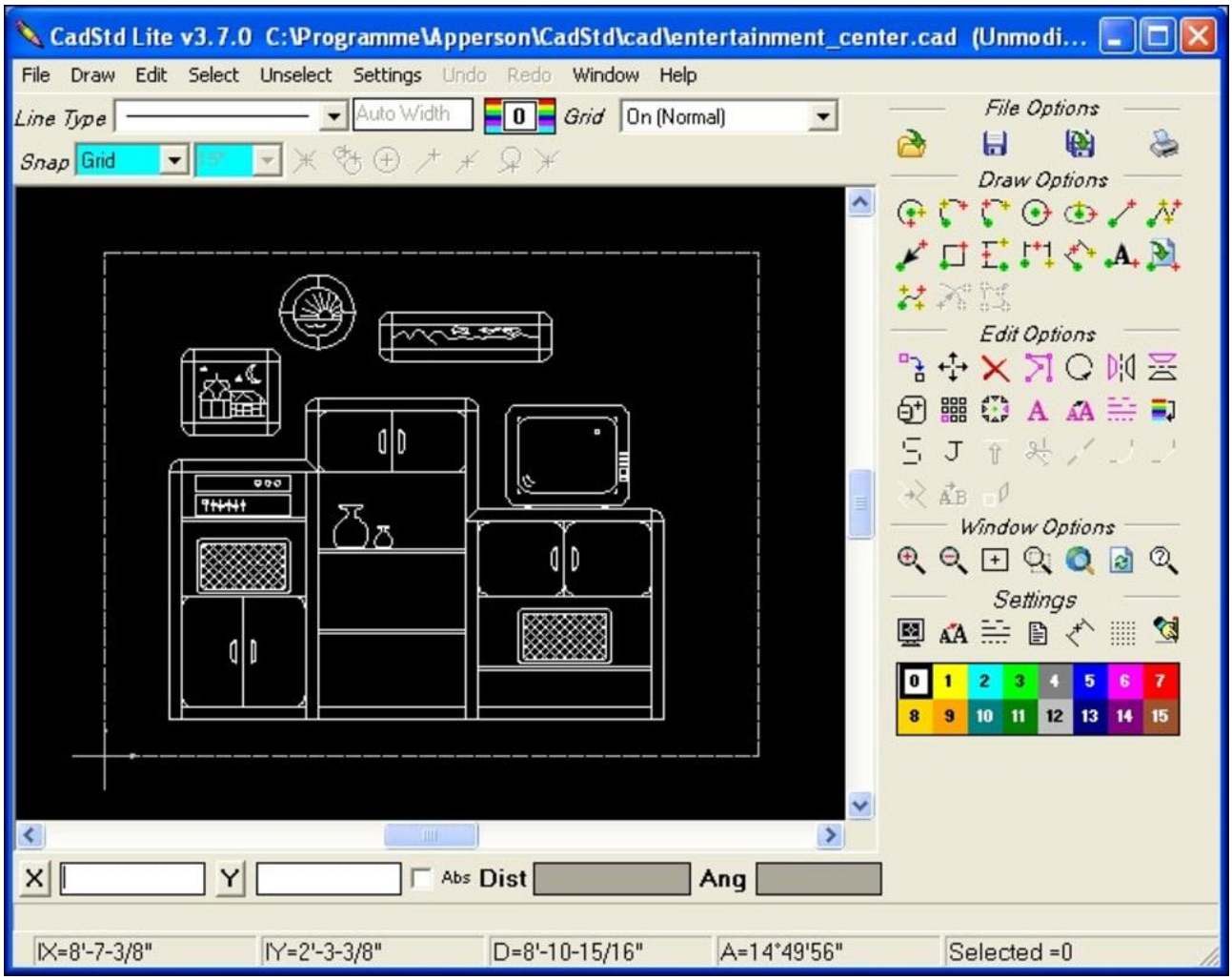

Рисунок 1.6. – Інтерфейс програми CadStd

1.7. CLOBBI

Це система, дозволяє планувати виробництво, та контролювати перебіг операцій, але основний його недолік, слабкий зв’язок з фізичною частиною виготовлення деталей. А саме, немає можливості напряму запускати виготовлення деталей на певних станках.

Тим не менше, все що стосується моніторингу роботи станкового парку, та безпосередньо планування його роботи реалізовано дуже грамотно.

Зокрема слід зауважити такі функції як [9]:

- розрахунок оптимальних планів виробництва. Clobbi дозволяє здійснювати контроль виробництва, при якому буде проводитися потрібну кількість

продукції за мінімальний час і з мінімальними витратами. Здійснюючи контроль виробництва продукції, Clobbi сформує змінне завдання кожному робітникові, а також видасть завдання прямо на мобільний телефон. Розрахунки плану виробництва наведено на рисунку 1.8;

- повна простежуваність партій у виробництві. Контроль на виробництві продукції від Clobbi дає можливість подивитися, який виробничий етап йде, а також проконтролювати відповідність партій і швидкості процесів виробництва вашим планам;
- 3D-цех - погляд в цех з будь-якої точки світу. Можна без хвилювань їздити у відрядження і на відпочинок - виробництво залишається на відстані смартфона або комп'ютера;
- Інтеграція з системами 3D-проектування. Можна швидко і легко перенести склад вироби з САПР в Clobbi, а це значно зменшить витрати часу на заповнення громіздких довідників ресурсів підприємства. Приклад роботи з моделями наведено на рисунку 1.7.

Оперативне планування виробництва і збуту продукції, мається на увазі короткостроковий деталізований розподіл всіх етапів роботи компанії. Його цілі полягають у доведенні конкретних виробничих завдань до кожного відділу, цеху, робочого ланки або бригади. Програма виробництва та реалізації продукції в такому типі ведення діяльності визначає розподіл однієї головної задачі на підзадачі на короткі тимчасові сегменти і встановлення планів на кожен з них.

Планування реалізації продукції в такому підході до виробничого процесу має на увазі ведення календарного і об'ємного планування, і диспетчеризація (контролю продажу продукції, що випускається фірмою продукції, управлінні цим процесом).

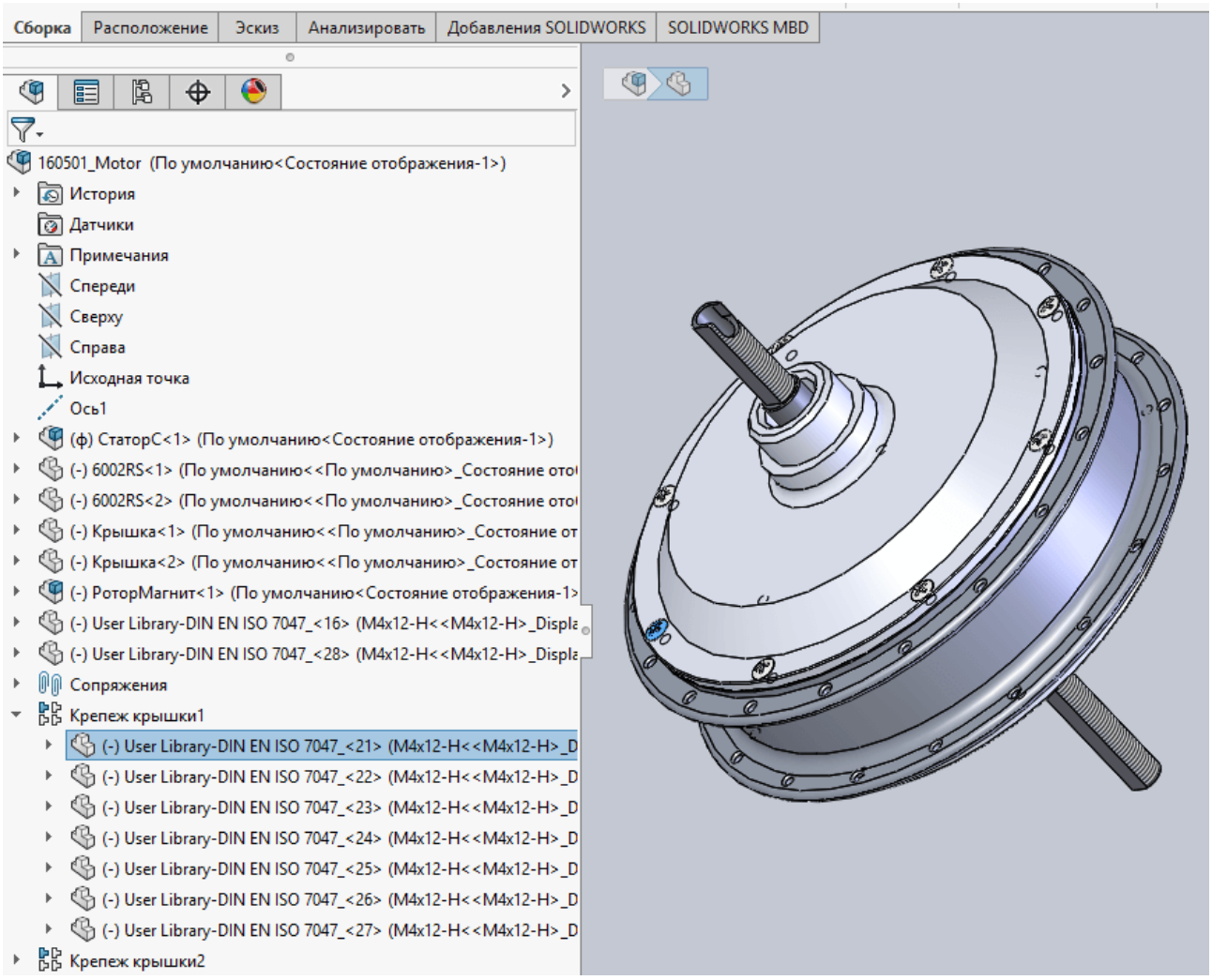

Рисунок 1.7. – Інтерфейс програми Clobbi, при роботі з моделями

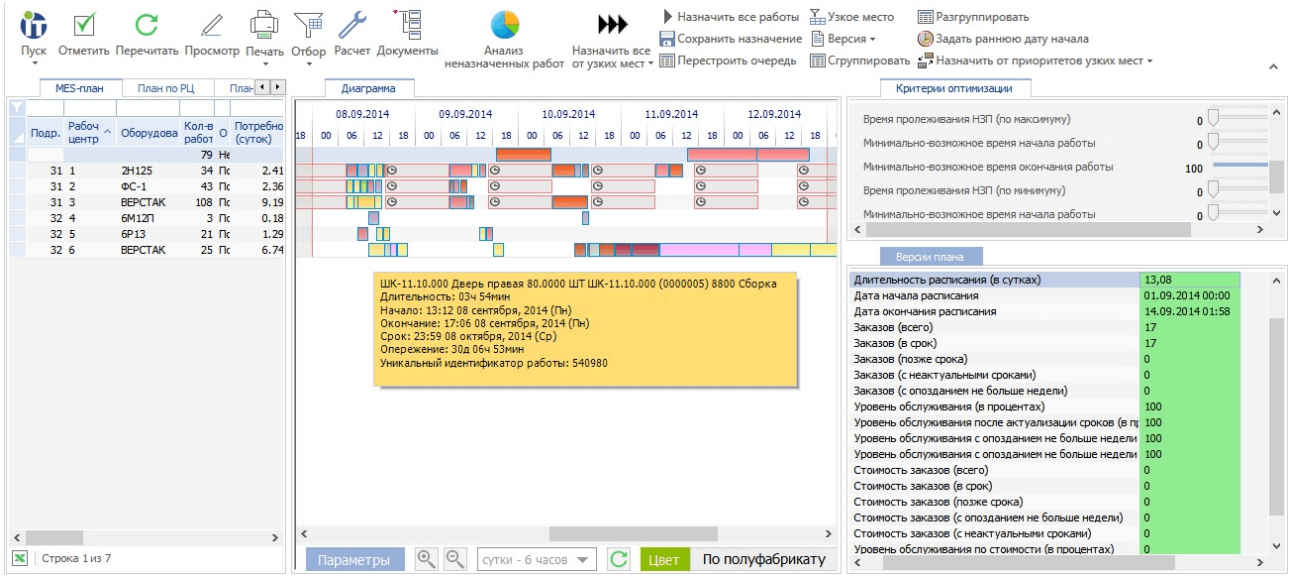

Рисунок 1.8. – Інтерфейс програми Clobbi, при режимі планування

Висновки до розділу 1

Як можна побачити програми що наведені в пунктах 1.1-1.6 відрізняються між собою багатьма параметрами: двовимірні чи тривимірні операції доступні, безкоштовні чи по ліцензії, деякі програми працюють лише з певними типами станків, а для деяких станків можливо використовувати тільки програмне забезпечення виробника.

Тому для вирішення поставленої задачі, немає сенсу використовувати якусь одну існуючу програму, або написати свою. Більш логічним рішенням буде можливість використовувати вже існуючі програми, і поєднувати їх з програмою-генератором розкладу роботи шляхом підключення певних бібліотек, що будуть відповідати за роботу з конкретними керуючими станками, і в випадку коли в групі станків є різні моделі, на них можливо буде встановлювати спеціалізовані програми керування.

Окремо слід звернути увагу на програму в пункті 1.7, так як вона найповноцінніша програма керування. Провівши огляд усіх аналогів, можна виокремити недоліки та переваги кожної окремої програми. Відповідно на основі цієї інформації можна створювати свою власну систему, та визначити можливі недоліки.

2. ФОРМУВАННЯ ВИМОГ ДО СИСТЕМИ ТА СЦЕНАРІЇВ ВИКОРИСТАННЯ

Формування вимог до системи, або цей процес ще називають аналіз вимог – це процес, що допомагає визначити, що саме очікують користувачі від нових програмних продуктів. Ці очікування користувачів, це і є вимоги, які зокрема і потрібно визначити, та розписати їх в усіх деталях. Під час розробки програмного забезпечення, аналіз вимог є дуже важливим етапом. Також управління вимогами це важливий аспект управління проектом.

На цій фазі розробки програмного продукту проводиться спілкування з цільовою аудиторією майбутньої системи, для того щоб визначити конкретні вимоги щодо основних функцій системи. Необхідно розв’язати усі можливі конфлікти що можуть виникнути, та прояснити всі можливі непорозуміння, що можуть виникнути між командою розробки та користувачами системи. Вимоги дозволяють прояснити весь майбутній процес розробки програмного забезпечення. Та коли вимоги були узгоджені, та задокументовані, це дозволяє уникнути конфліктів на завершальному етапі розробки.

Вимоги поділяються на функціональні і нефункціональні вимоги. Аналіз вимог можливо поділити на три етапи [10]:

- збір вимог, в свою чергу в себе включає огляд існуючих рішень та спілкування з цільовою аудиторією;
- аналіз вимог – це визначення усіх основних характеристик майбутньої системи. Перевірки вимог, пошук можливих конфліктів між вимогами;
- документація, в якій записуються усі узгоджені з замовником вимоги, бажано офіційно затвердити її.

2.1. Функціональні вимоги

Функціональні вимоги (FR - functional requirements) – це вимоги, що покликані забезпечити функціональність програмного продукту. Функціональність системи залежить від безпосередньо поведінки системи та вихідних і вхідних параметрів.

Також потрібно прописати взаємодію даних в системі, взаємодію з оператором, та інший специфічний функціонал[10].

Функціональні вимоги:

- система має приймати на вхід дані від зовнішніх програм (зокрема програм безпосереднього управління станками з числовим програмним управлінням);
- система повинна мати можливість прийняти на вхід інформацію з текстового файлу, або електронних таблиць;
- система повинна зберігати дані у спеціальному сховищі даних з чіткою структурою, при чому окремо слід зауважити, що сховище даних повинно мати високий рівень захищеності;
- система повинна обробляти слабкоструктуровані дані, що надходять від оператора, та на їх основі генерувати розклад роботи виробництва;
- система повинна зберігати усі дані, що система генерує під час своєї роботи (стани станків, кількість виготовлених деталей, стан готовності замовлення) в базу даних;
- система повинна реалізовувати механізми генерування розпорядку роботи групи станків з ЧПУ;
- система повинна мати можливість використовувати зовнішні бібліотеки, що будуть використовуватись для роботи з зовнішніми програмами для керування ЧПУ станками;
- система повинна легко розповсюджуватись через мережу Інтернет;
- система повинна мати окрему програму, що буде відповідати за оновлення компонентів програми, та розгортання системи;
- система повинна мати можливість зберігати специфічні файли, що відповідають за моделі деталей, які були створені в спеціальних програмах, та відправляти ці моделі в програми безпосереднього управління станками з ЧПУ;

2.2. Нефункціональні вимоги

Нефункціональні вимоги (NFR - nonfunctional requirements) – вимоги, що визначають властивості, які система повинна демонструвати, або ж обмеження котрі система повинна виконувати, при цьому ці обмеження не повинні безпосередньо відноситись до поведінки системи. Наприклад можуть визначатись такі параметри як: продуктивність, зручність підтримки, надійність, фактори що можуть виникати під час експлуатації, розширюваність [10].

Нефункціональні вимоги:

- система повинна генерувати розклад не довше ніж за 180 секунд роботи;
- система повинна оброблювати групи станків не більше 20 одиниць;
- система повинна перевіряти формати файлів моделей;
- система повинна бути реалізовуватись по ціні, щоб бути конкурентоспроможною близько 50 умовних одиниць;
- система має мати можливість імпортувати дані через API;
- система має бути покрита тестами на 100%, та усі варіанти можливих помилок мають бути опрацьовані та ні в якому випадку не приводити до зупинки системи;
- система повинна працювати на операційній системі сімейста ОС Windows;
- система повинна використовувати не більше ніж 512 мегабайт оперативної пам`яті;
- можливість роботи з зовнішніми ПУ по API;

2.3. Ролі користувачів в системі

Користувачів системи можна поділити на два типи це оператор (працівник що працює з системою, відповідає також за введення інформації даних і моделей в систему та виведення інформації з системи); та користувач що являється адміністратором системи. Можна побачити на діаграмі в додатку 9.

Функції користувачів:

1) Оператор – відповідає за попередній аналіз даних, що вводяться в систему (завдання на партії деталей, інформація про станки, моделі необхідні для створення деталей). Може вносити корективи в безпосередній процес виробництва (наприклад зупинити якийсь конкретний странок);
2) Адміністратор – отримує усю готову інформацію по розкладу роботи, якщо потрібно адміністратор власноруч може особисто внести в розклад корективи, адміністратор так само отримує повну статистичну інформацію по роботі групи станків з ЧПУ. Відповідно він так само отримує інформацію по загальним витратам на матеріали та можливий прибуток підприємства. Також адміністратор має усі ті самі можливості по управлінню системою, що й оператор.

2.4. Опис сценаріїв використання

Детальний опис сценаріїв використання представлено в таблицях 2.1-2.6 Діаграма діяльності наведена в додатку 7.

Таблиця 2.1. – Опис сценарію використання «Внесення даних у систему»

Назва сценарію	Внесення даних у систему
Причина події	Внесення даних користувачем, що спричиняє їх попередній аналіз
Опис	Для початку процесу внесення даних до системи, потрібно щоб оператор вручну заповнив відповідні текстові поля, або запустив зчитування з файлу в якому збережено інформацію.
Частота події	Може відбутись в будь який момент часу.
Актори	Оператор, Адміністратор

Продовження таблиці 2.1

Попередні умови	Оператор повинен бути зареєстрованим адміністратором у системі, та має авторизуватись в ній.
Попередні умови	Дані введені оператором повинні пройти попередню верифікацію в системі, для уникнення ситуацій коли можливо ввести хибну інформацію (наприклад текст в поле де повинно бути число)
Вихідні умови	Дані оброблюються системою, та зберігаються в базі даних, для подальшої їх обробки, та (коли це необхідно) аналізу.
Опис дій	1. Оператор вносить в систему певну вхідну інформацію (не так важливо що конкретно вводить оператор, інформацію про партію деталей що потрібно виготовити, чи наприклад це буде специфікація до нового станка в системі, або модель певної деталі) в відповідні поля, що відповідають даним. 2. Оператор натискає кнопку «Зберегти». 3. Система перевіряє введені користувачем дані. 4. Якщо дані введені коректно система зберігає їх у відні таблиці в базі даних. Якщо в введеній користувачем інформації є помилки, система виводить відповідне повідомлення, та повертається в стан 1. пункту.
Очікувані результати	Створено відповідний запис у базі даних, збереження введеної оператором інформації

Таблиця 2.2. – Опис сценарію використання «Автоматична генерація розкладу роботи системою для оператора»

Назва сценарію	Автоматична генерація розкладу роботи системою
Причина події	Оператор або адміністратор запустили процес генерації розкладу роботи групи станків з ЧПУ
Опис	На основі даних що збережені в базі даних системи, система генерує попередній варіант розкладу роботи, якщо необхідно, адміністратор може внести корективи в розклад роботи, оператор вносити корективи в розклад не може, після узгодження розкладу, він зберігається в базі даних системи.
Частота події	Може відбутись в будь який момент часу.
Актори	Оператор
Попередні умови	Оператор повинен бути зареєстрованим адміністратором у системі, та має авторизуватись в ній. В базі даних системи повинна бути наявна інформація про партію чи партії деталей що повинні бути виготовлені, та інформація про наявні і доступні промислові ресурси (кількість, можливості, та продуктивність усіх доступних станків з ЧПУ)
Вихідні умови	Остаточний результат генерації розкладу роботи (безпосередньо розклад) зберігається до бази даних системи.

Продовження таблиці 2.2

Опис дій	1. Оператор або адміністратор запускають процес генерації розкладу, натискаючи кнопку «згенерувати розклад». 2. Система перевіряє чи достатньо в базі даних інформації для генерації розкладу роботи, якщо достатньо переходить до пункту 3, якщо ні – виводить відповідне повідомлення. 3. Система на основі евристичного алгоритму знаходить певне проміжне допустиме рішення (генерується проміжний варіант розкладу). 4. Результат отриманий в результаті кроку 3, використовується для реалізації алгоритму «Гілок і Меж» в якості початкового «рекорду», та відбувається процес оптимізації початкового рішення. 5. Результат роботи кроку 4 виводиться в новому вікні в вигляді вже згенерованого розкладу. 6. Розклад зберігається до бази даних
Очікувані результати	Створено відповідний запис у базі даних, збереження файлу розкладу.

Таблиця 2.3. – Опис сценарію використання «Автоматична генерація розкладу роботи системою для адміністратора»

Назва сценарію	Автоматична генерація розкладу роботи системою
Причина події	Оператор або адміністратор запустили процес генерації розкладу роботи групи станків з ЧПУ

Продовження таблиці 2.3

Опис	На основі даних що збережені в базі даних системи, система генерує попередній варіант розкладу роботи, якщо необхідно, адміністратор може внести корективи в розклад роботи, оператор вносити корективи в розклад не може, після узгодження розкладу, він зберігається в базі даних системи.
Частота події	Може відбутись в будь який момент часу.
Актори	Адміністратор
Попередні умови	Адміністратор має бути авторизованим у системі. В базі даних системи повинна бути наявна інформація про партію чи партії деталей що повинні бути виготовлені, та інформація про наявні і доступні промислові ресурси (кількість, можливості, та продуктивність усіх доступних станків з ЧПУ)
Вихідні умови	Остаточний результат генерації розкладу роботи (безпосередньо розклад) зберігається до бази даних системи.
Опис дій	1. Адміністратор запускають процес генерації розкладу, натискаючи кнопку «згенерувати розклад». 2. Система перевіряє чи достатньо в базі даних інформації для генерації розкладу роботи, якщо достатньо переходить до пункту 3, якщо ні – виводить відповідне повідомлення. 3. Система на основі евристичного алгоритму знаходить певне проміжне допустиме рішення (генерується проміжний варіант розкладу).

Продовження таблиці 2.3

Опис дій	4. Результат отриманий в результаті кроку 3, використовується для реалізації алгоритму «Гілок і Меж» в якості початкового «рекорду», та відбувається процес оптимізації початкового рішення. 5. Результат роботи кроку 4 виводиться в новому вікні в вигляді вже згенерованого розкладу. 6. Адміністратор може внести певні корективи в розклад. 7. Після внесення коректив (якщо вони були) розклад зберігається до бази даних
Очікувані результати	Створено відповідний запис у базі даних, збереження файлу розкладу.

Таблиця 2.4. – Опис сценарію використання «Запуск процесу роботи»

Назва сценарію	Запуск процесу роботи
Причина події	Оператор або адміністратор запустили процес виготовлення партії або партій деталей на групі станків з ЧПУ
Опис	На основі розкладу, що було створено в попередніх сценаріях використання відбувається виготовлення партій деталей на групі станків з ЧПУ.
Частота події	Може відбутись в будь який момент часу.
Актори	Адміністратор, Оператор
Попередні умови	Адміністратор має бути авторизованим у системі. Оператор повинен бути зареєстрованим адміністратором у системі, та має авторизуватись в ній. В базі даних системи повинен бути згенерований розклад роботи. Усі станки робота яких запланована в розкладі, мають бути доступні для програми.

Продовження таблиці 2.4

Вихідні умови	Після завершення роботи усіх станків програма перевіряє їх стан, звертаючись по API до програм що безпосередньо відповідають за роботу кожного конкретного станка, та виводить звіт про перебіг виконання розкладу.
Опис дій	1. Адміністратор або оператор запускають процес роботи над партіями деталей, вибравши зі списку відповідний розклад, що було згенеровано раніше, або було завантажено окремо, та натискаючи кнопку «розпочати роботу». 2. Система перевіряє чи усі станки що задіяні в розкладі доступні, якщо так переходить до наступного кроку, якщо ні – виводить відповідне повідомлення. 3. Система перевіряє чи не працюють станки в даний час, якщо працюють пропонує внести корективи в розклад, якщо ні – переходить до наступного пункту. 4. Система для кожного станка формує групи деталей що потрібно виконати (можна уявити як певні доріжки, поділені на відрізки, кожен відрізок – певна деталь), та готує файли моделей для кожної деталі (дістає файли з бази даних в оперативну пам’ять). 5. Система відправляє програмам управління кожним конкретним станком моделі деталей. 6. Коли станок повідомляє про завершення деталі, система відправляє йому модель наступної.

Продовження таблиці 2.4

	7. Коли на всіх станках завершено виконання партій деталей, система опитує кожну керуючу програму по API про стан станка. 8. На основі опитування генерується звіт про виконання розкладу. Звіт зберігається в базі даних.
Очікувані результати	Файл звіту, групи готових деталей, та запис у базі даних.

Таблиця 2.5. – Опис сценарію використання «Генерація статистики»

Назва сценарію	Генерація статистики
Причина події	Адміністратор натискає відповідну кнопку в меню
Опис	На основі інформації про ціну групи заготовок, відомостей про станки (кількість електроенергії що споживається кожним конкретним станком, знос свердл та самого станка), ціни готових виробів, адміністратор отримує інформацію доходи та витрати.
Частота події	Може відбутись в будь який момент часу.
Актори	Адміністратор
Попередні умови	Адміністратор має бути авторизованим у системі. В базі даних системи повинен бути згенерований розклад роботи. Або повинен бути файл звіту про виконання розкладу.

Продовження таблиці 2.5

Вихідні умови	Система виводить аналітичний звіт по економічним показникам, можна побачити всю інформацію про роботу над певними замовленнями, та зберегти файл аналітичного звіту.
Опис дій	1. Адміністратор вибирає розклад, або звіт виконання розкладу на основі якого потрібно згенерувати файл статистики.
	2. Адміністратор натискає кнопку «Згенерувати» 3. Система на основі розкладу або файлу звіту робить підрахунок витрат та доходів в ході роботи над партією деталей. 4. Система виводить статистичну інформацію на екран, та зберігає її в базі даних. Адміністратор може вибрати в якому форматі буде збережено файл статистики.
Очікувані результати	Файл статистики по витратам та доходам під час роботи над певним розкладом.

Таблиця 2.6. – Опис сценарію використання «Моніторинг роботи станків з ЧПУ»

Назва сценарію	Моніторинг роботи станків з ЧПУ
Причина події	Запущено виконання певного розкладу
Опис	Відбувається постійний обмін інформацією між системою, та програмами безпосереднього керування станками з ЧПУ по API.
Частота події	Відбувається постійно під час роботи над розкладом
Актори	Система
Попередні умови	Запущено виконання певного розкладу, і станки з ЧПУ вже почали свою роботу.

Продовження таблиці 2.6

Вихідні умови	Система записує стан кожного станка в певний момент часу. (Під станом мається на увазі, операція яку в данний момент виконує станок, скоріше всього запис буде зроблено в вигляді G-коду.
Опис дій	1. Система звертається до програми управління конкретним станком з ЧПУ. 2. Система отримує відповідь про стан станка в поточний момент (яку операцію він виконує). 3. Система записує цю інформацію в базі даних.
Очікувані результати	Система записує в базі даних, усю інформацію про те що робить кожен станок в кожен конкретний момент часу.

Висновки до розділу 2

Опис сценаріїв взаємодії задовольняє вимогам, що ставилися до програмно-системного комплексу, які були наведені на початку даного розділу магістерської дисертації. На основі представлених сценаріїв, розробляються тести, що перевіряють роботу системи на коректність та стійкість. На основі сценаріїв використання і таблиць 2.1-2.6 наведених вище будується діаграма прецендентів та діяльності. Наступним етапом розробки системи є визначення структурної схеми системи.

3. СТРУКТУРНА СХЕМА СИСТЕМИ

Призначення структурної схеми це відображення компонентів та взаємодія різних частин системи між собою. Дана схема зазвичай демонструє присутність в системі певних підсистем, а також різних компонентів, що в свою чергу складають кінцеву систему, та забезпечують взаємодію та управління між різними рівнями системи.

Важливим етапом розробки програмного забезпечення є розробка різних схем, що допоможуть зрозуміти загальну структуру програми, та спростити проектування системи. Зокрема структурна схема включає в себе набір компонентів з яких безпосередньо й складається система. По схемі можна простежити взаємозв’язки між цими компонентами, та додаткові компоненти що використовуються в ході роботи системи. Маючи перед очима структурну схему, та її похідні варіанти можна легко сформувати уявлення про систему, та це в свою чергу спростить внесення виправлень та модифікацій у систему в майбутньому. Також це полегшує масштабування системи, якщо в подальшому буде поставлена така задача.

Система що представлена в магістерській дисертації побудована на принципах концепції багаторівневої архітектури. Не дивлячись на те, що можна зустріти багато різних типів архітектури, на практиці застосовують зазвичай звичайну трирівневу систему, що дозволяє розбити програму на три рівні.

Слід окремо зауважити, що використання рівнів в архітектурі часто пов’язано з такими поняттями як n-шар та n-ярус. В зв’язку з тим що ці поняття дещо схожі, їх доволі часто плутають.

Ярус являє собою певний фізичний рівень, на якому працює певна частина системи, іншими словами наприклад клієнт-серверний додаток має два шари, а саме дві програми що фізично рознесені на різних машинах, або є можливість їх фізично рознести.

В той час як шар являє собою більш логічний розподіл того як працює програма, не обов’язково навіть вона має бути розподіленою. Є різни рівні доступу до інформації, суть в тому щоб користувач не міг напряму впливати на те що збережено

в базі даних. Є інтерфейс користувача, котрий реалізовує ввід інформації до системи, та вивід інформації користувачу, так званий рівень бізнес логіки, рівень на якому відбувається уся логічна робота програми, перевірка введених даних, та їх обробка. Та рівень безпосередньо даних, з яким і відбувається взаємодія для збереження даних.

Рисунок 3.1. – Схема трирівневої архітектури

На рисунку 3.1. зображено схему трирівневої архітектури. Варто пам’ятати що крайні шари не можуть напряму взаємодіяти один з одним, тобто рівень відображення даних не може напряму звертатись до рівня даних, а робить це через рівень бізнес логіки. Структурну схему наведено в додатку 6.

3.1. Діаграма компонентів

Діаграма компонентів (component diagram) описує особливості фізичного представлення системи. Діаграма компонентів дозволяє визначити архітектуру розроблюваної системи, встановивши залежності між програмними компонентами, в ролі яких може виступати вихідний, бінарний і виконуваний код. У багатьох

середовищах розробки модуль або компонент відповідає файлу. Пунктирні стрілки, що з'єднують модулі, показують відносини взаємозалежності, аналогічні тим, які мають місце при компіляції вихідних кодів програм. Основними графічними елементами діаграми компонентів є компоненти, інтерфейси і залежності між ними.

Діаграма компонентів розробляється для наступних цілей:

- візуалізація загальної структури вихідного коду програмної системи;
- специфікація виконуваного варіанти програмної системи;
- забезпечення багаторазового використання окремих фрагментів програмного коду;
- представлення концептуальної і фізичної схем баз даних.

Як можна побачити на наведеній на рисунку 3.2 діаграмі система складається з машини на якій встановлено програмний комплекс генерації розкладу, та набору станків з ЧПУ, в яких на контролерах встановлено програми що безпосередньо відповідають за керування станком. Комп’ютер оператора та група станків з ЧПУ знаходяться в локальній мережі, в якій і відбувається основна їх взаємодія між собою.

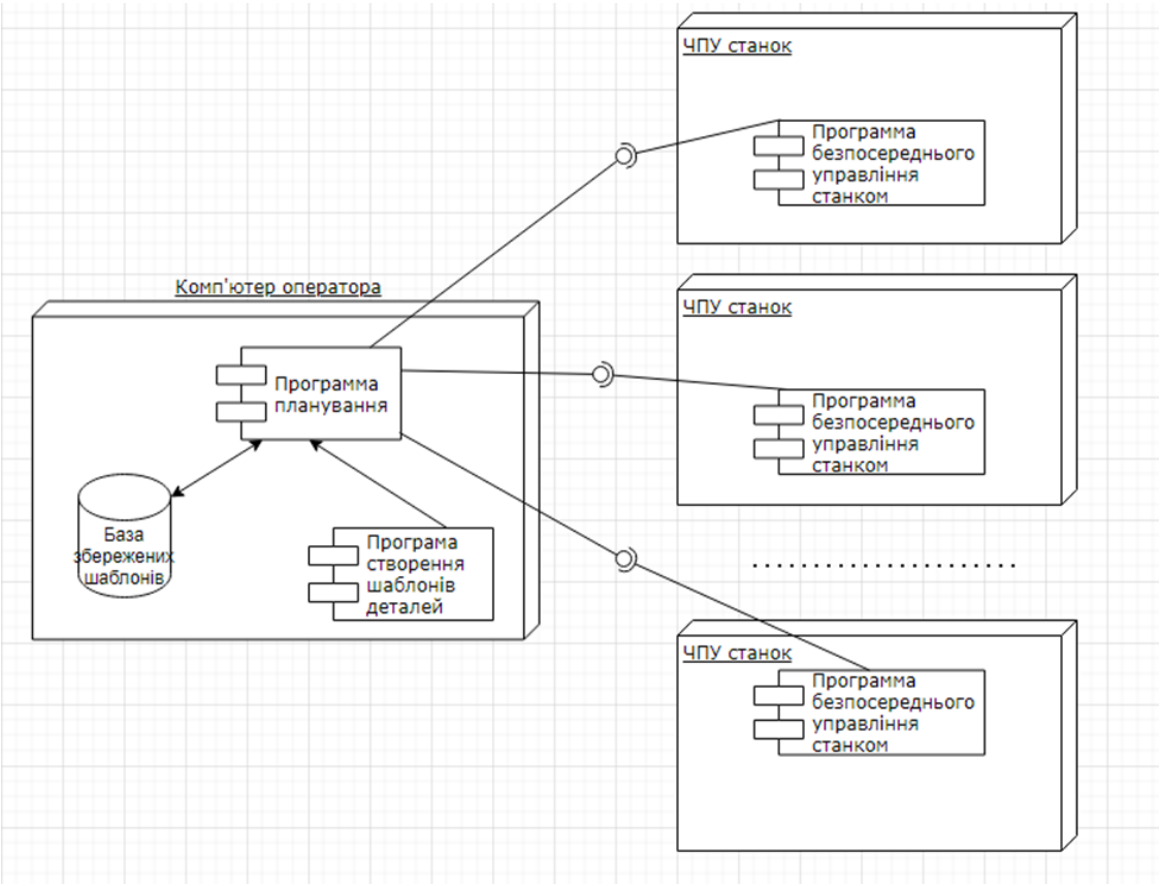

Рисунок 3.2. – Діаграма компонентів системи

3.2. Опис функціональної моделі

Опис функціоналу що було реалізовано у системі наведено нижче у форматі діаграми IDEF0 (рис.3.3 - рис. 3.6). Ці діаграми наведено в додатках 1-2.

Методологія IDEF0 передбачає побудову ієрархічної системи діаграм – одиничних описів фрагментів системи. Спочатку проводиться опис системи в цілому і її взаємодія з оточуючим світом (рис. 3.3), після чого проводиться функціональна декомпозиція – система розбивається на підсистеми і кожна підсистема описується окремо (рис. 3.4 – рис 3.6). Після чого кожна з підсистем в свою чергу може бути розбита на менші підсистеми, і так далі, доки не буде досягнуто потрібного рівня деталізації.[11]

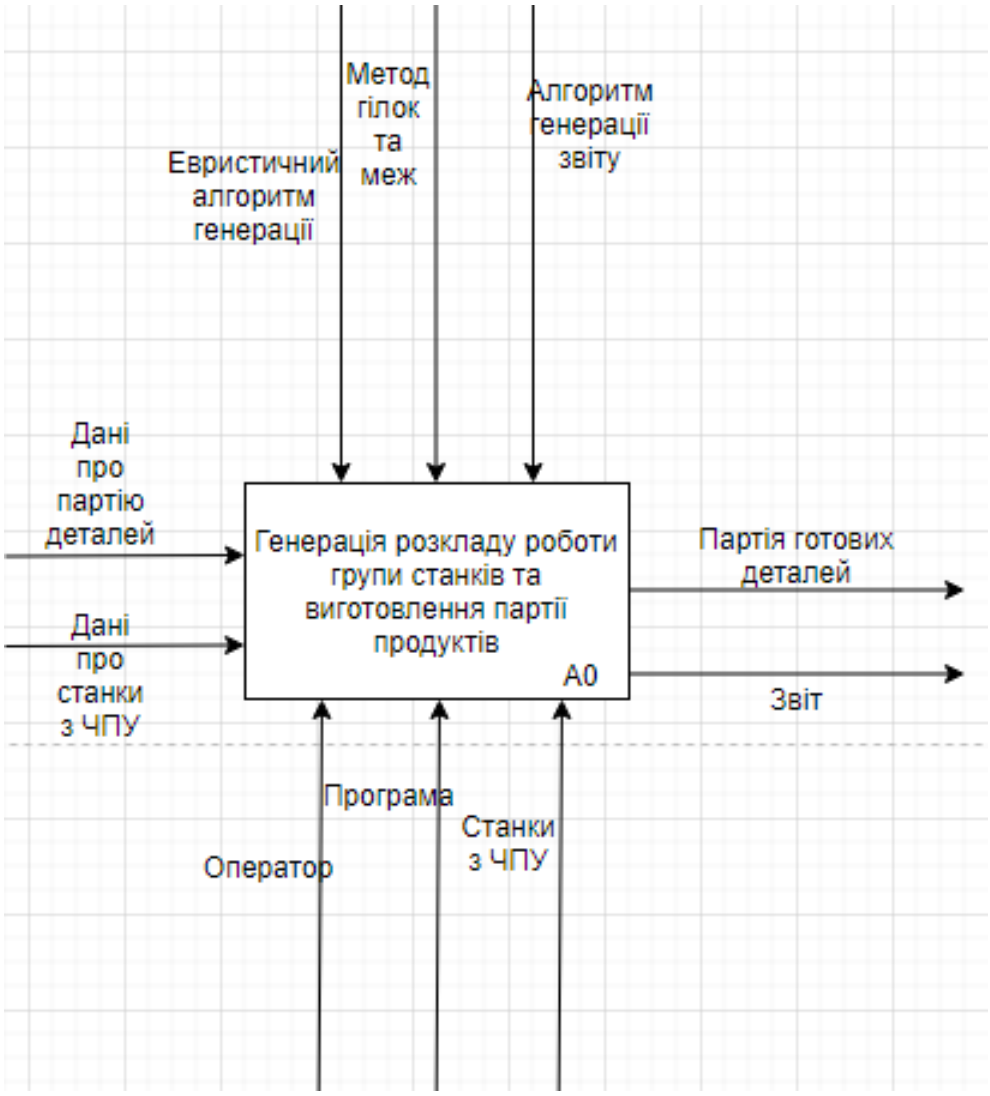

Рисунок 3.3 – Схема структурна контекстної моделі системи

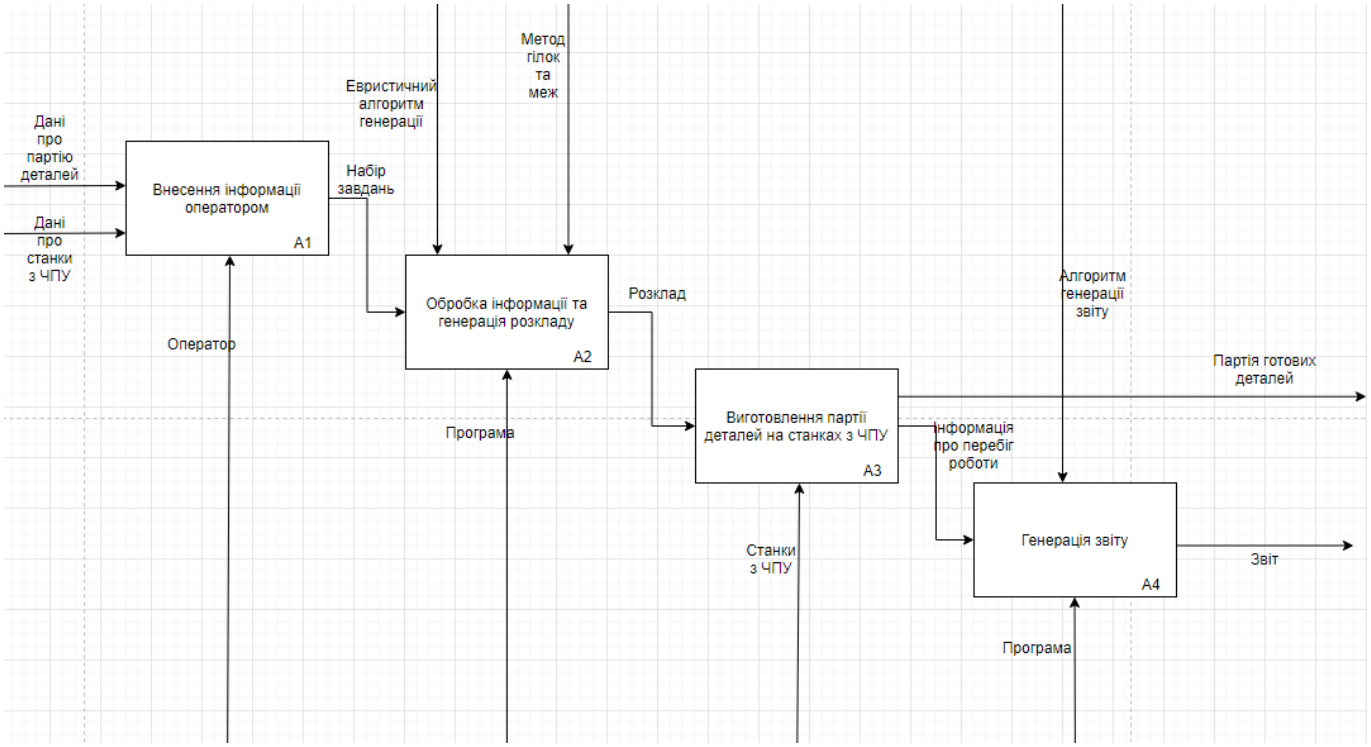

Рисунок 3.4 – Схема структурна функціональної моделі IDEF0

На наведеній на рисунку 3.3 діаграмі відображено лише один блок – це головна функція нашої системи, а саме створення розкладу роботи групи станків з ЧПУ на основі якого в подальшому буде виконано створення партії готових деталей (або виробів). Усе це відбувається на основі внесеної оператором інформації про станки з ЧПУ, та безпосередньо інформації про деталі, які в подальшому мають бути виготовлені.

На рисунку 3.4 наведено перший рівень декомпозованої системи, головна функція системи управління групою станків з ЧПУ(Генерація розкладу роботи групи станків та виготовлення партії продуктів) була розбита на 4 задачі:

- внесення інформації користувачем;
- обробка інформації та генерація розкладу;
- виготовлення партії деталей на станках з ЧПУ;
- генерація звіту.

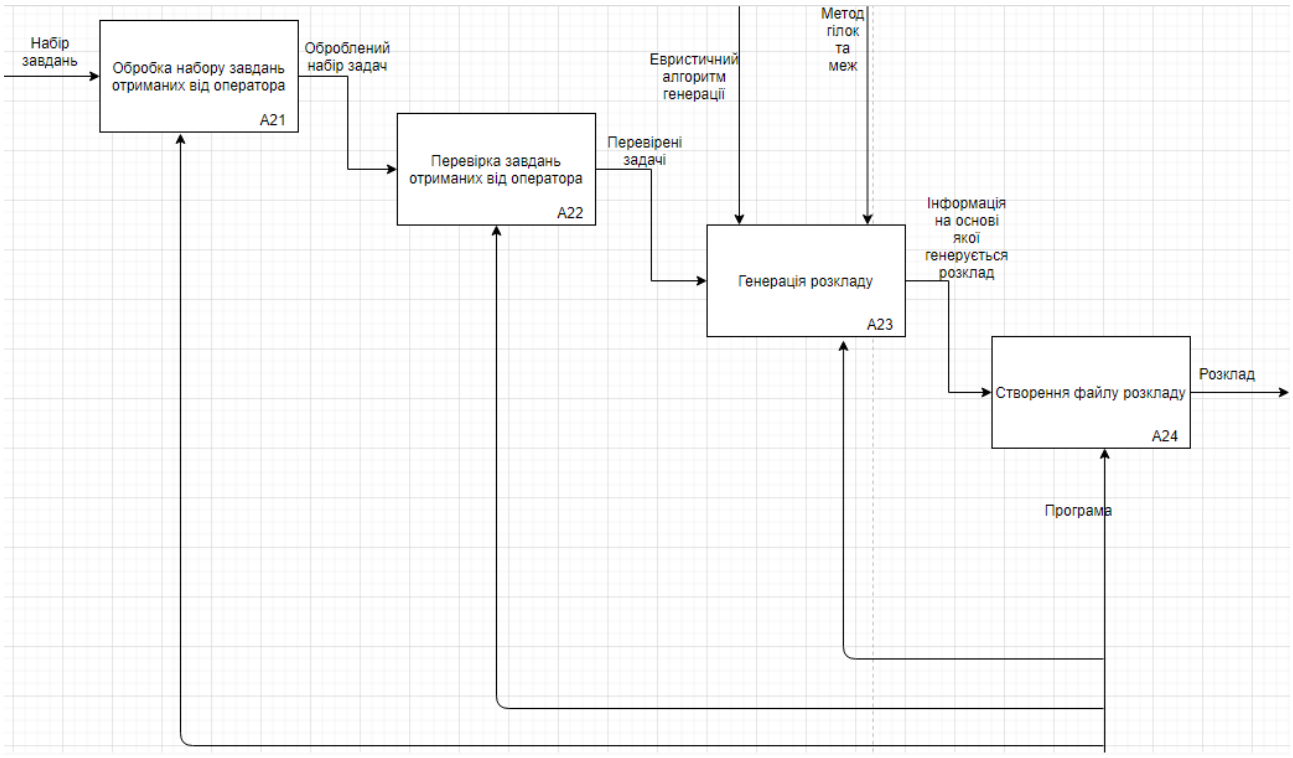

Рисунок 3.5 – Схема структурна функціональної моделі IDEF0

На рисунку 3.5 зображено декомпозицію функції що відповідає за генерацію розкладу на основі якого в подальшому буде створено готові деталі. Показано чотири підзадачі:

- обробка набору завдань отриманих від оператора;
- перевірка завдань отриманих від оператора;
- генерація розкладу;
- створення файлу розкладу.

На рисунку 3.6 зображено декомпозицію функції генерації звіту по перебігу виконання розпорядку. Основна задача (генерація звіту) була розділена на три підзадачі:

- структурування інформації про перебіг роботи;
- генерація звіту на основі структурованої інформації;
- створення файлу звіту.

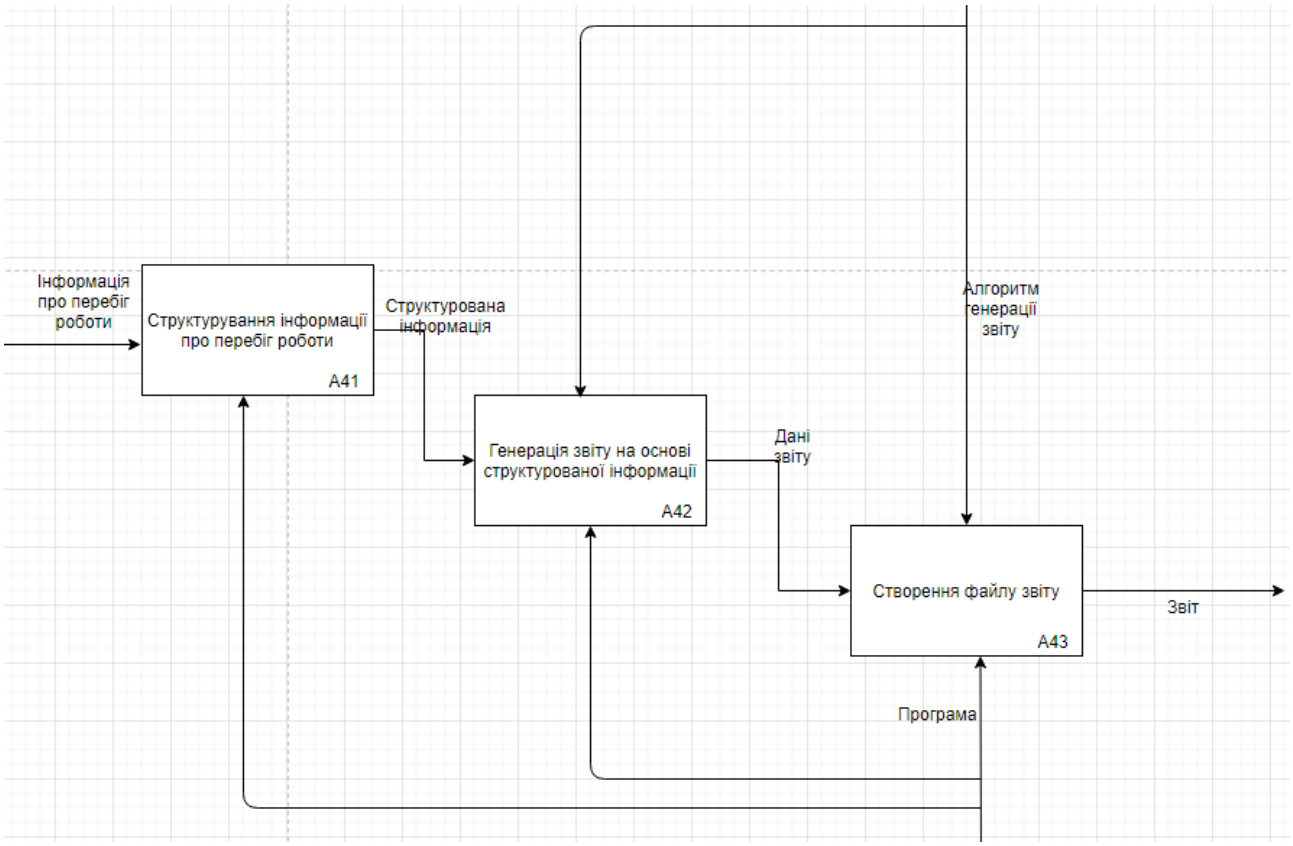

Рисунок 3.6 – Схема структурна функціональної моделі IDEF0

3.3. Основні підсистеми

Система це певний набір модулів, або підсистем, що певним чином взаємодіють між собою. Перша підсистема відповідає за роботу з даними, що надходять ззовні системи. На вхід надходять дані в відповідному вигляді (або через поля вводу користувачів, або відповідно через обробку файлів зі структурованою інформацією. Ця підсистема зчитує вхідні дані, проводить їх певну первинну обробку, та передає інформацію на вхід підсистеми безпосередньо відповідальної за обробку даних (рис 3.7). Також ця система отримує на вхід дані про поточний стан підключених станків з ЧПУ, на основі яких в подальшому буде створено звіт по виконанню розкладу роботи над групою деталей. Повна діаграма в додатку 3.

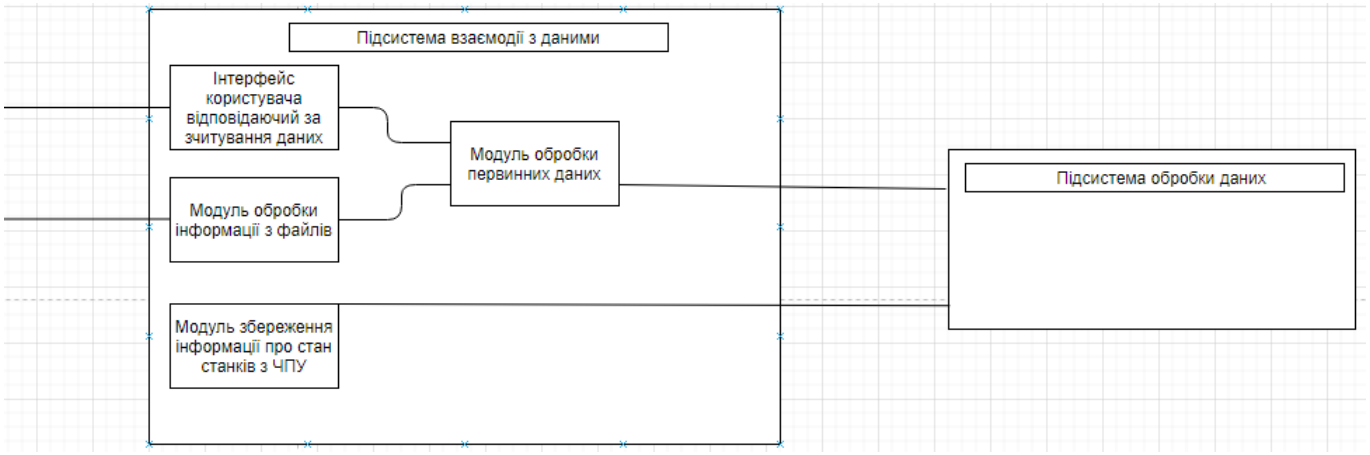

Рисунок 3.7 – Схема взаємодії з даними

Далі йде рівень що безпосередньо відповідальний за безпосередню обробку даних (рис 3.8). Вона включає в себе модулі, що відповідальні за обробку даних, за генерацію розкладу, за генерацію звіту, та модуль відповідальний за збереження результатів у системі. В модулі обробки використовуються бібліотеки класів, що відповідають за створення певного структурованого набору даних, модулі генерації розкладу та звітів це складні набори класів, що на основі структурованого набору вхідних даних створюють файли розкладу та звіту, модуль збереження інформації відповідає за зберігання всієї інформації в системі, він видаляє всю зайву інформацію, що могла бути в даних в попередніх етапах (різні прапорці, проміжні результати, тощо), та передає інформацію в модуль зберігання даних.

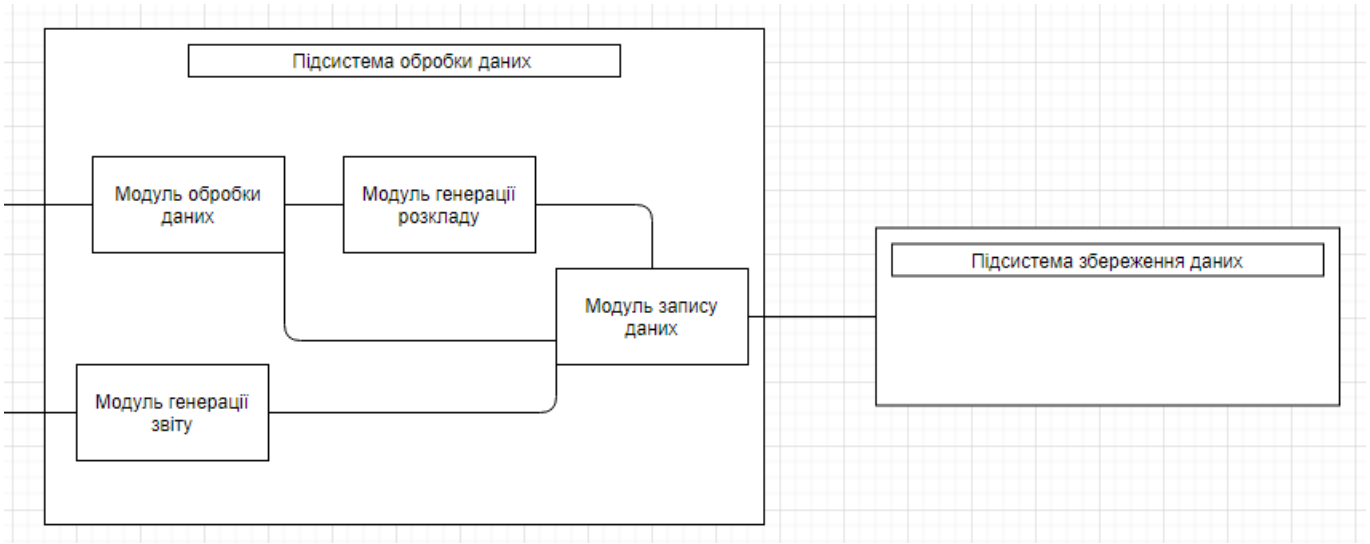

Рисунок 3.8 – Схема підсистеми обробки даних

Наступний рівень це рівень збереження даних (рис 3.9). На цьому рівні знаходиться два елементи, модуль що відповідає за валідацію та збереження інформації (реалізовано на базі фреймворку), та база даних, в якій і зберігаються усі дані системи, вона представляє з себе набір структурованих таблиць, в яких і робляться записи.

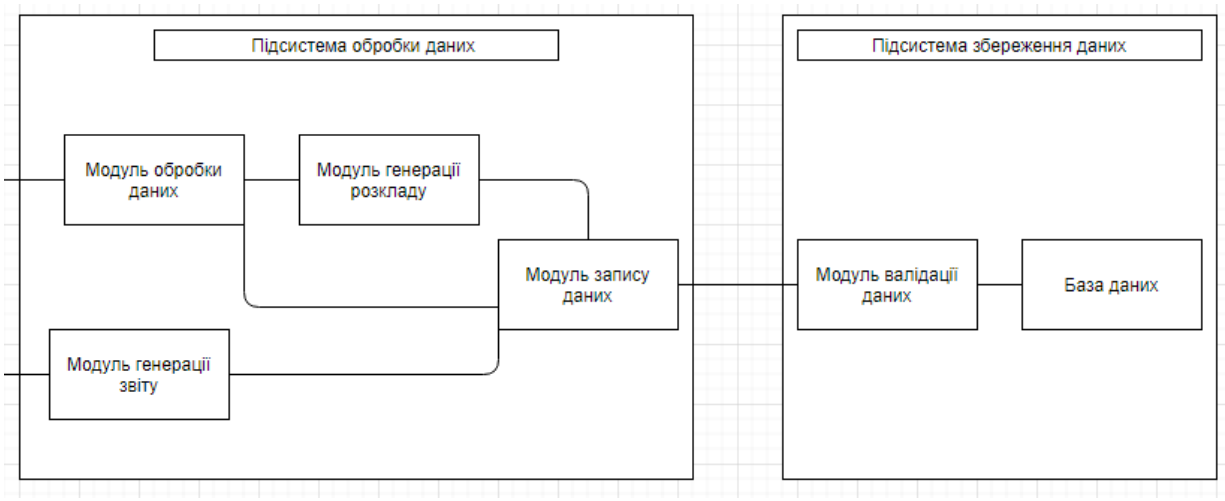

Рисунок 3.9 – Схема підсистеми збереження даних

Висновки до розділу 3

В представленій у магістерській дисертації системі доцільно буде використовувати багаторівневу архітектуру, так як використовується декілька різних логічних модулів, які при цьому можуть бути на різних рівнях. Даний підхід до створення архітектури системи дозволить полегшити розгортання системи, і також надасть інструментарій для подальшого масштабування програмної системи, якщо таке завдання виникне в майбутньому. Також наявна можливість модифікувати існуючі або додавати нові рівні до нашої системи без великих ускладнень. Визначившись з архітектурою системи можливо переходити до наступного етапу розробки системи. Наступний етап це вибір технологій, що будуть використані в ході розробки системи.

4. ВИБІР ТА ОБГРУНТУВАННЯ ЕЛЕМЕНТІВ ТА ТЕХНОЛОГІЙ

Важливий етап проектування розробки програмного забезпечення, це визначення тих технологій що будуть використані в ході безпосереднього створення системи. Так як система має різні рівні, це дозволяє використовувати різні технології на різних рівнях, але тим не менш важливо вибрати ці технології заздалегідь, структурувати їх, та визначитись з повним переліком стеку технологій що будуть використовуватись. Змінити технології в ході безпосередньої розробки системи звісно можливо, але для цього доведеться витратити дуже багато часу, на те щоб замінити певну технологію, та навчити систему заново з нею працювати. Існує багато обмежень по взаємодії різних технологій між собою, тож необхідно усі ці взаємозв’язки заздалегіть спланувати.

4.1. Вибір платформи розробки системи

На основі даних, що були отримані в попередньому розділі, а саме вимог до системи, було вирішено обрати платформу .NET 5.0 для розробки модулів системи, що реалізують елементи інтеграції та платформу .NET Framework для модулів, що будуть відповідати за обробку тих даних, що виникають під час роботи системи [12].

Нижче наведено детальнішу інформацію про ці платформи.

У 2002 році корпорація Microsoft випустила .NET Framework, платформу розробки для створення додатків Windows. Сьогодні .NET Framework має версію 4.8 і продовжує підтримуватися корпорацією Microsoft.

У 2014 році корпорація Microsoft почала створювати кросплатформенну альтернативу .NET Framework з відкритим вихідним кодом. Ця нова реалізація .NET аж до версії 3.1 називалася .NET Core. Наступна версія після .NET Core 3.1 - це .NET 5.0, яка в даний час знаходиться на стадії попередньої версії. Версія 4 була пропущена, щоб уникнути плутанини між цією реалізацією .NET і .NET Framework 4.8. Назва «Core» була упущена, для того щоб прояснити, що тепер це основна реалізація .NET [12].

.NET 5.0 – це наступний великий крок у дереві життєвого циклу платформи .NET Core після версії 3.1. Назва нової версії .NET 5.0 замість .NET Core 4.0 була вибрана по двум причинам [12]:

- було вирішено пропустити номери версій 4.x, для того щоб уникнути плутань з .NET Framework 4.x;
- слово «Core» було виключено з назви, для того щоб підкреслити що це основний напрямок розвитку платформи .NET на майбутнє. І .NET 5.0 підтримує більшу кількість типів додатків та платформ ніж .NET Core або .NET Framework;

Підсумовуючи .NET це безкоштовна платформа розробки з відкритим кодом для створення різних додатків та програм, таких як зокрема:

- веб додатки;
- додатки що працюють в хмарі;
- мобільні додатки;
- десктопні додатки такі як: Windows WPF, Windows Forms та UWP;
- інтернет речей;
- консольні додатки;
- служби Windows.

Реалізована можливість спільного використання функцій між різними додатками, та навіть різними типами додатків з допомогою бібліотек класів. З .NET код програм та файли проекту виглядають однаково, незалежно від того, який саме додаток ви створєюєте. Завжди є доступ до одного й того ж середовища виконання, API та мовним можливостям кожного окремого додатку, що ви створюєте.

Основні плюси використання платформи .NET 5.0 [12]:

- Кросплатформеність.

Можливість створювати додатки .NET для багатьох операційних систем, включаючи:

1) Linux
2) Android
3) macOS

4) iOS
5) tvOS
6) Windows
7) watchOS

Більше того платформа підтримує навіть різні архітектури процесорів такі як:

1) x64
2) x86
3) ARM32
4) ARM64

При цьому .NET дозволяє використовувати специфічні для платформи можливості, такі як API операційної системи. Прикладами є Windows Forms і WPF в Windows і власні прив'язки до кожної мобільній платформі від Xamarin.

- Відкритий код.

.NET - це відкритий вихідний код, який використовує ліцензії MIT і Apache 2. .NET - це проект, що розробляється та підтримується .NET Foundation.

- Підтримка.

.NET підтримується компанією Microsoft в Windows, macOS і Linux. Він регулярно оновлюється для забезпечення безпеки та якості, буквально кожного місяця виходять нові оновлення. Двійкові дистрибутиви .NET від компанії Microsoft створюються і тестуються на підтримуваних Microsoft серверах в Azure і відповідають методам розробки та безпеки, що встановлені в самій корпорації Microsoft.

- Можливість писати код програми на різних мовах програмування, зокрема підтримуються три мови програмування: C#, F#, Visual Basic.

Зокрема програма буде написана на мові програмування C#. C# (вимовляється як «See Sharp») - це сучасна, об'єктно-орієнтована і типозахищена мова програмування. Мова C # бере свій початок в сімействі мов C і буде відразу непогано знайома знаком програмістам що писали код на таких популярних і частозустрічаючихся мовах програмування як C, C ++, Java і JavaScript.

- Менеджер пакетів.

NuGet - це менеджер пакетів з відкритим вихідним кодом, розроблений для платформи .NET. Пакет NuGet - це файл архіву .zip з .nupkgpa розширенням, який містить скомпільований код бібліотек (DLL), інші файли, пов'язані з цим кодом, і описовий маніфест, який включає таку інформацію, як номер версії пакета. Розробники створюють пакети і публікують їх на сайті nuget.org або на приватному хості. Розробники, які хочуть використовувати чужий доступний код, додають пакет в свій проект і потім можуть викликати API, що надається пакетом в своєму коді проекту. Документацію по кожному конкретному пакету можна знайти зазвичай на офіційному сайті розробника коду.

- CLR

.NET надає середовище виконання коду, що називається загальномовне виконуюче середовище, в котрому виконується код і де надаються служби, що спрощують розробку. Детально це описано на рисунку 4.2.

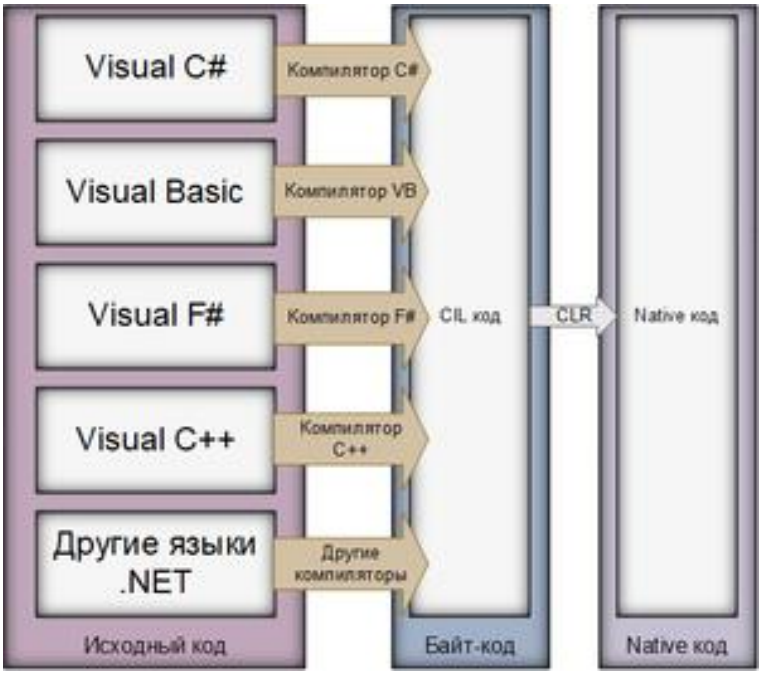

Рисунок 4.2 – Виконання коду з CLR на різних мовах

Компілятори надають інструменти, що дають використовувати функціональні можливості загальномовного середовища, та дозволяють писати код програм, який вигідно відрізнятиметься від коду скомпільованого звичайним компілятором. Код

котрий генерується за допомогою компілятору та назначений для середовища виконання називають керуючим кодом. Керуючий код дозволяє отримати такі переваги як загальномовна обробка виключень, підвищена безпека коду, підтримка можливостей керування версіями та розгортанням додатків, спрощена модель взаємодії компонентів в системі, а також послуги налагодження та профілювання.

Щоб середовище виконання могло надавати послуги керуючому коду, мовні компілятори повинні повертати метадані, що описують типи, члени та посилання в коді програміста. Метадані зберігаються поряд з файлами коду. Кожен виконуваний файл середовища CLR має в собі метадані. Середовище виконання використовує метадані для пошуку та завантаження класів, розміщення екземплярів об'єктів у пам'яті, дозвіл на виклики методів, генерація власного коду, забезпечення безпеки та встановлення меж контексту та часу виконання [13].

Середовище виконання автоматично оброблює макет об'єкту і керує посиланнями на об'єкти, звільняючи пам'ять від цих посилань, коли об'єкти більше не використовуються. Об'єкти, час життєвого циклу котрих регулюється таким чином, називають керуючими даними. Збір сміття позбавляє нас проблеми надмірного споживання пам'яті, а також може нас врятувати від ще кількох частих помилок в програмуванні. Якщо ваш код являється керованим, ви можете використовувати керуючі дані, дані якими не можете керувати, або ж використовувати як керовані дані, так і некеровані дані в своєму додатку на платформі .NET. Оскільки компілятори мов надають користувачу свої власні типи, такі як примітивні типи, ви не завжди можете знати (або не завжди ви повинні знати), керовані ваші дані чи ні.

Середовище CLR спрощує розробку бібліотек та додатків вцілому, для об'єктів котрі взаємодіють з різними мовами. Об'єкти написані на різних мовах програмування, можуть взаємодіяти один з одним, і їх поведінка може бути тісно інтегрована в рамках одного додатку. Наприклад ви на одній мові можете створити клас, а потім використати іншу мову для наслідування цього класу з вихідного класу, або викликати метод вихідного класу на іншій мові. Ви також можете передавати екземпляри класів написаних на одній мові, в якості параметрів методів що написані

на іншій мові програмування. Ця загальномовна інтеграція можлива оскільки компілятори мов використовують систему загальних типів, що була визначена в середовищі виконання, і вони слідують правилам встановленим середовищем виконання для створення нових користувацьких типів, а також для створення, використання, зберігання та прив’язки до типів [13].

В складі своїх метаданих усі керуючі компоненти несуть інформацію про ресурси і компоненти, для котрих вони і були створені. Середовище виконання використовує цю інформацію для того щоб генерувати ваш компонент або додаток в певній версії, з використанням усіх ресурсів що йому потрібні для роботи, що в свою чергу знижує ймовірність поломки вашої програми через наявність якоїсь залежності, для якої немає відповідного їй файлу. Реєстраційна інформація та дані про стан більше не зберігаються в реєстрі де їх важко створювати та підтримувати. Замість цього інформація про ті типи що ви створюєте та їх залежності зберігається разом з кодом в вигляді метаданих, що значно спрощує задачі реплікації і видалення компонентів.

Мовні компілятори та їх інструменти розкривають функціональні можливості середовища виконання способами, котрі мають бути корисними та інтуїтивно зрозумілими для розробників. Це означає, що деякі функції середовища можуть бути в деяких випадках більш наглядно помітними ніж в інших. Те як ви сприймаєте середовище виконання залежить від того які мовні компілятори або інструменти ви використовуєте. Середовище виконання дає наступні переваги:

1) підвищення продуктивності;
2) можливість легкого інтегрування компонентів розроблених на інших мовах;
3) розширювані типи, що представлені різними бібліотеками класів на різних мовах;
4) функції мови, такі як інтерфейси, успадкування і перевантаження для об'єктно-орієнтованого програмування;
5) підтримка явною вільної потокової передачі, яка дозволяє створювати багатопотокові масштабовані додатки;
6) підтримка структурованої обробки виключень;

7) підтримка налаштовуваних атрибутів;
8) звільнення від сміття;
9) використовування делегатів замість показників функцій для підвищення безпеки типів;

- Автоматичне керування пам’яттю.

Збирач сміття (GC) управляє розподілом і виділенням пам'яті для додатків. Кожен раз, коли ваш код створює новий об'єкт, середовище CLR виділяє для цього об'єкту пам'ять з керованої купи. Поки адресний простір є в керованій купі, виконавче середовище продовжує виділяти простір для нових об'єктів. Коли залишається недостатньо вільного адресного простору, GC перевіряє об'єкти в керованій купі, які більше не використовуються додатком. Потім він відновлює цю пам'ять.

GC - одна з служб середовища CLR, яка допомагає забезпечити безпеку пам'яті. Програма безпечна для пам'яті, якщо вона звертається тільки до виділеної пам'яті. Зокрема, виконавче середовище гарантує, що програма не отримає доступ до нерозподіленої пам'яті за межами, наприклад, масиву.

- Робота з некерованими ресурсами.

Іноді код повинен посилатися на некеровані ресурси. Некеровані ресурси - це ресурси, що автоматично не обслуговуються середовищем виконання .NET. Наприклад, дескриптор файлу - це некерований ресурс. Об'єкт FileStream - це керований об'єкт, але він посилається на дескриптор файлу, який не є керованим. Коли ви закінчите використовувати FileStream, вам потрібно явно звільнити дескриптор файлу.

В .NET об'єкти, які посилаються на некеровані ресурси, реалізують інтерфейс IDisposable. Коли ви закінчите використовувати об'єкт, ви викликаєте метод об'єкта Dispose(), який відповідає за звільнення будь-яких некерованих ресурсів. Окрім цього в .NET є зручний оператор using, який можна використовувати для здійснення методу Dispose.

4.2. Вибір технології розробки

Оскільки програма буде мати графічний інтерфейс, було вирішено використовувати технологію WPF (Windows Presentation Foundation) котра являється частиною екосистеми платформи .NET, та являє собою підсистему для створення додатків з графічним інтерфейсом. Тож саме цю технологію було вирішено використовувати для розробки системи.

Якщо при створенні традиційних додатків на основі WinForms за відображення елементів керування і графіку відповідали такі частини ОС Windows як User32 і GDI+, то в додатках WPF основна задача лягає на DirectX. В цьому й заключається основна перевага й особливість рендерингу графіки в WPF: використовуючи WPF значна частина роботи по відображенню графіки, як найпростіших кнопок , так і складних 3D-моделей, лягає на графічний процесор на відеокарті, що в свою чергу дозволяє скористатись апаратним прискоренням графіки [14].

Одна з найважливіших особливостей являється використання мови декларативної розмітки інтерфейсу XAML, яка в свою чергу засновується на XML: ви можете створювати насичений графічний інтерфейс, використовуючи або декларативне ініціалізування інтерфейсу, або код на керуючих мовах програмування C#, або ви можете взагалі сувміщати ці два підходи до створення інтерфейсу.

Основні переваги WPF [14]:

1) створення додатків під ОС сімейства Windows - від Windows XP до Windows 10;
2) апаратне прискорення графіки - незалежно від того, чи працюєте ви з 2D або 3D, графікою або текстом, всі компоненти програми транслюються в об'єкти, зрозумілі Direct3D, і потім візуалізуються за допомогою процесора на відеокарті, що підвищує продуктивність та робить графіку більш плавною;
3) багаті можливості по створенню різних додатків: це і мультимедіа, і двомірна і тривимірна графіка, і багатий набір вбудованих елементів управління, а також можливість самим створювати нові елементи, створення анімацій, прив'язка даних, стилі, шаблони, теми і багато іншого;

4) хороша взаємодія з WinForms, завдяки чому, наприклад, в додатках WPF можна використовувати традиційні елементи управління з WinForms;
5) нові можливості, яких складно було досягти в WinForms, наприклад, створення тривимірних моделей, прив'язка даних, використання таких елементів, як стилі, шаблони, теми і ін;
6) незалежність від розширення екрану: оскільки в WPF всі елементи вимірюються в незалежних від пристрою одиницях, додатки на WPF легко масштабуються під різні екрани з різним розширенням;
7) можливість декларативного визначення графічного інтерфейсу за допомогою спеціальної мови розмітки XAML, заснованому на xml і представляє альтернативу програмному створення графіки та елементів управління, а також можливість комбінувати XAML і C # / VB.NET;
8) використання традиційних мов .NET-платформи - C # і VB.NET для створення логіки додатка;

У той же час WPF має певні обмеження. Незважаючи на підтримку тривимірної візуалізації, для створення додатків з великою кількістю тривимірних зображень, перш за все ігор, краще використовувати інші засоби - DirectX або спеціальні фреймворки, такі як Monogame або Unity.

Також варто враховувати, що в порівнянні з додатками на Windows Forms обсяг програм на WPF і споживання ними пам'яті в процесі роботи в середньому трохи вище. Але це з лишком компенсується більш широкими графічними можливостями і провишенной продуктивністю при відображенні графіки. Тож для розробки програми було вибрано саме WPF.

Як можна побачити на рисунку 4.3, WPF розбивається на два рівні, managed API і unmanaged API (рівень інтеграції з DirectX). Managed API (керований API-інтерфейс) включає в себе код, що виконується під керуванням загальномовного виконуючого середовища .NET - Common Language Runtime.

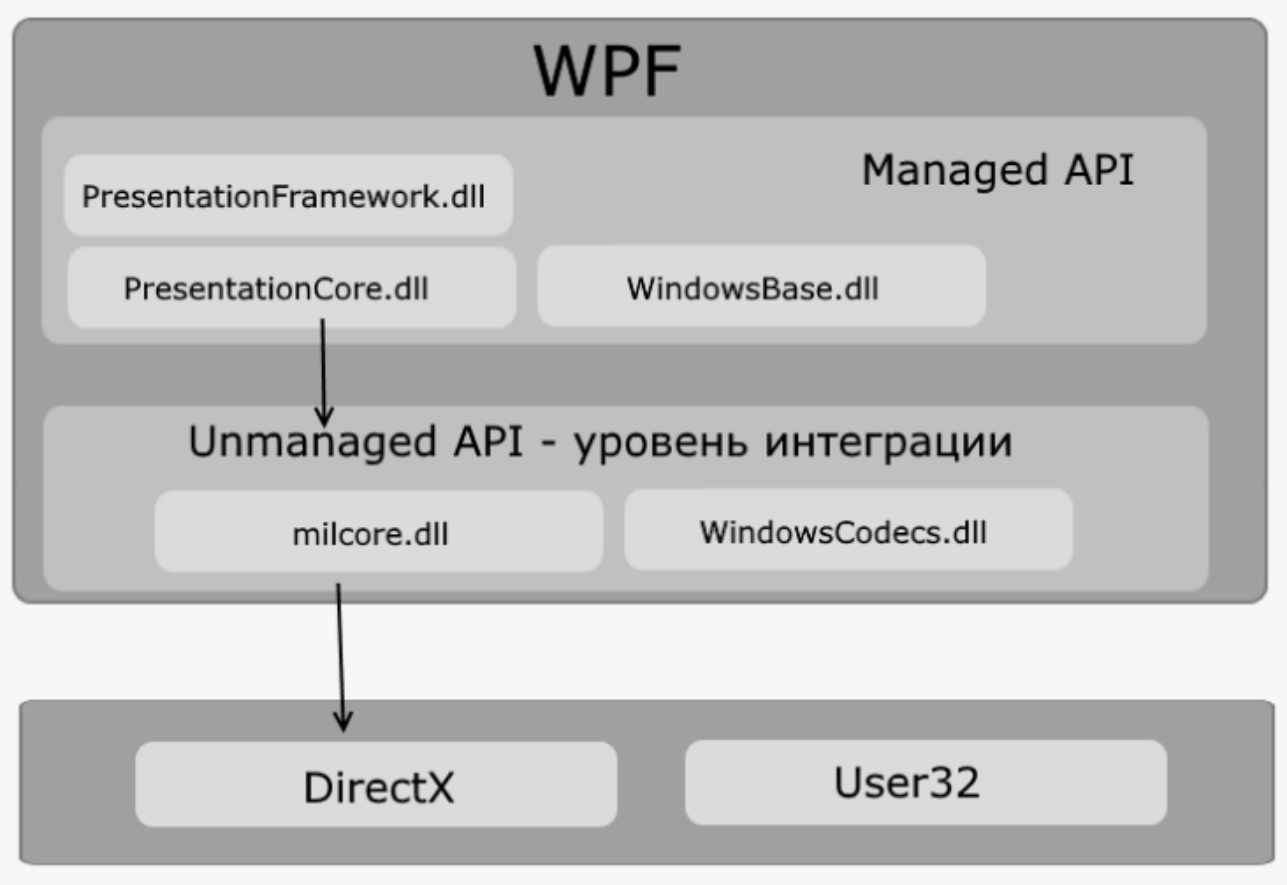

Рисунок 4.3 – Схема архітектури WPF

Цей API описує основний функціонал платформи WPF і складається з наступних компонентів [14]:

- PresentationFramework.dll: містить всі основні реалізації компонентів і елементів управління, які можна використовувати при побудові графічного інтерфейсу;
- PresentationCore.dll: містить всі базові типи для більшості класів з PresentationFramework.dll;
- WindowsBase.dll: містить ряд допоміжних класів, які застосовуються в WPF, але можуть також використовуватися і поза даної платформи;

Unmanaged API використовується для інтеграції вищележащого рівня в DirectX:

- milcore.dll: власне забезпечує інтеграцію компонентів WPF з DirectX. Даний компонент написаний на некерованому коді (C / C ++) для взаємодії з DirectX;

- WindowsCodecs.dll: бібліотека, яка надає низькорівневу підтримку для зображень в WPF;

Ще нижче знаходяться власне керуючі компоненти операційної системи та DirectX, котрі й дозволяють візуалізувати компоненти додатку, або виконують іншу низькорівневу обробку. Зокрема, за допомогою низькорівневого інтерфейсу Direct3D, який входить до складу DirectX, відбувається трансляція.

Тут також на одному рівні знаходиться бібліотека user32.dll. І хоча вище говорилося, що WPF не використовує цю бібліотеку для рендеринга і візуалізації, проте для ряду обчислювальних задач (що не включають візуалізацію) дана бібліотека продовжує використовуватися.

4.3. Бази даних

Для роботи з базою даних в системі було вирішено використовувати технологію Entity Framework Core (EF Core). Вона являє собою об'єктно-орієнтовану, легковажну і расширяемую технологію від компанії Microsoft для доступу до даних. EF Core є ORM-інструментом (object-relational mapping - відображення даних на реальні об'єкти). Тобто EF Core дозволяє працювати базами даних, але є більш високий рівень абстракції: EF Core дозволяє абстрагуватися від самої бази даних і її таблиць і працювати з даними незалежно від типу сховища. Якщо на фізичному рівні оперуємо таблицями, індексами, первинними і зовнішніми ключами, але на концептуальному рівні, який пропонує Entity Framework, вже працюємо з об'єктами [15].

Entity Framework Core підтримує безліч різних систем баз даних. Таким чином, можемо через EF Core працювати з будь-якої СУБД, якщо для неї є потрібний провайдер [15].

За замовчуванням на даний момент Microsoft надає ряд вбудованих провайдерів: для роботи з MS SQL Server, для SQLite, для PostgreSQL. Також є провайдери від сторонніх постачальників, наприклад, для MySQL.

Також варто відзначити, що EF Core надає універсальний API для роботи з даними. І якщо, наприклад, ми вирішимо змінити цільову СУБД, то основні зміни в проекті будуть стосуватися насамперед конфігурації і настройки підключення до

відповідних провайдерам. А код, який безпосередньо працює з даними, отримує дані, додає їх в БД і т.д., залишиться колишнім.

Entity Framework Core багато успадкував від своїх попередників, зокрема, Entity Framework 6. У той же час треба розуміти, що EF Core - це не нова версія по відношенню до EF 6, а зовсім інша технологія, хоча в цілому принципи роботи у них будуть збігатися . Тому в рамках EF Core використовується своя система версій. Поточна версія - 5.0 була випущена в листопаді 2020 року. І технологія продовжує розвиватися [15].

Як технологія доступу до даних Entity Framework Core може використовуватися на різних платформах стека .NET. Це і стандартні платформи типу Windows Forms, консольні додатки, WPF, UWP і ASP.NET Core. При цьому кроссплатформенная природа EF Core дозволяє задіяти її не тільки на ОС Windows, але і на Linux і Mac OS X [15].

Центральної концепцією Entity Framework є поняття сутності або entity. Сутність визначає набір даних, які пов'язані з певним об'єктом. Тому дана технологія передбачає роботу не з таблицями, а з об'єктами і їх колекціями.

Будь-яка сутність, як і будь-який об'єкт з реального світу, має низку властивостей. Наприклад, якщо сутність описує людини, то можемо виділити такі властивості, як ім'я, прізвище, зріст, вік. Властивості необов'язково представляють прості дані типу int або string, але можуть також представляти і більш комплексні типи даних. І у кожної сутності може бути одна або кілька властивостей, які будуть відрізняти цю сутність від інших і будуть унікально визначати цю сутність. Подібні властивості називають ключами.

При цьому суті можуть бути пов'язані асоціативної зв'язком один-ко-многим, один-ко-одному і багато-до-багатьох, подібно до того, як в реальній базі даних відбувається зв'язок через зовнішні ключі.

Відмінною рисою Entity Framework Core, як технології ORM, є використання запитів LINQ для вибірки даних з БД. За допомогою LINQ можемо створювати різні запити на вибірку об'єктів, в тому числі пов'язаних різними асоціативними зв'язками.

А Entity Framework при виконання запиту транслює вираження LINQ в вирази, зрозумілі для конкретної СУБД (як правило, в вирази SQL).

Висновки до розділу 4

На основі вимог, що були висунуті до кінцевої системи, було визначено які компоненти мають бути реалізовані в системі.

Отже для створення програми буде використовуватись технологія Windows Presentation Foundation або скорочено WPF. Саме ця технологія дозволить створити інтерфейс системи, що буде функціональним, та при цьому зможе легко масштабуватись не залежно від розширення екрану конкретного користувача. Технологія Windows Presentation Foundation являється частиною платформи .NET 5.0.

Зокрема на платформі .NET 5.0 будуть створюватися бібліотеки класів, в яких буде реалізовано функціонал системи, дана модульна конструкція дозволить легко транспортувати готову систему на інші платформи, все що для цього буде потрібно реалізувати інтерфейс користувача на конкретній платформі, як тільки це буде зроблено загальномовне виконуюче середовище дозволить легко підключити вже існуючі бібліотеки роботи з станками з ЧПУ, бібліотеки генерації розкладів, та бібліотеки генерації звіту по завершенню робочого процесу.

Для реалізації роботи з базою даних можна використовувати Entity Framework Core 5.0. Що дозволить значно спростити взаємодію з систумою управління базою даних.

5. ER-ДІАГРАМА

В попередньому розділі, враховуючи висунуті вимоги до програмного продукту та на основі збору інформації про існуючі аналоги, та на основі аналізу доступних технологій, що можна використати, для роботи з базами даних було обрано Entity Framework Core 5.0.

Entity Framework Core 5.0 може працювати з багатьма видами баз даних. Для цього використовують так звані провайдери, провайдер відповідно це певний клас, що дозволяє нам відкинути фізичний рівень роботи з базою даних, та працювати з нею як зі звичайними об’єктами класів. Зокрема в Entity Framework Core має ряд вбудованих провайдерів, що дозволяють працювати з такими СУБД як MS SQL Server, SQLite, PostgreSQL. Також є провайдери, що надаються різними сторонніми розробниками, наприклад один з найпопулярніших аналогів це MySQL.

Ще один неймовірний плюс, це те що технології доступу до даних Entity Framework Core може використовуватись на різних платформах стеку .NET. Це і стандартні платформи типу Windows Forms, консольні додатки, WPF, UWP та ASP.NET Core. При цьому кросплатформенна природа EF Core дозволяє використовувати його не лише на операційній системі Windows, але й на Linux і Mac OS.

Для роботи програми було вирішено використовувати саме СУБД, що вже згадували, а саме MS SQL Server.

Для того щоб Entity Framework Core 5.0 успішно працював з потрібною нам базою даних необхідно підключити два пакети, через менеджер пакетів NuGet:

- Microsoft.EntityFrameworkCore.SqlServer – пакет що має в собі провайдер що відповідає за роботу з конкретною СУБД, а саме з MS SQL Server.
- Microsoft.EntityFrameworkCore.Tools

Після чого потрібно лише створити клас що відповідає за роботу з базою даних, або так званий клас контексту, який має бути наслідником класу DbContext. Для роботи з цим класом обов’язково було підключити такий простір імен як

Microsoft.EntityFrameworkCore. Серед всього набору класів цього простору імен слід виділити наступні:

- DbContext: визначає контекст даних, використовується для взаємодії з базою даних;
- DbSet/DbSet<TEntity>: представляє набір об’єктів, що зберігається в базі даних;
- DbContextOptionsBuilder: клас що відповідає за безпосереднє налаштування підключення.

Тим не менше маючі на увазі сценарії використання системи, було встановлено наступні сутності:

- Users;
- CNC_Machines;
- CNC_Machines_Parks;
- Jobs;
- Orders;
- Timetables;
- Reports;
- Prices;
- Programs;

До складу таблиці 5.1 «Users» входять наступні поля, і відповідно ця таблиця відповідає за вхід в програму, та розподіляли права між адміністратором та операторами:

- ім’я користувача;
- логін користувача;
- пароль користувача;
- права адміністратора;
- історія авторизацій та дій.

Таблиця 5.1 – «Users»

Код	Опис	Тип даних	Обов'язкове	Унікальне	Первинний ключ	Зовнішній ключ
Id	Ідентифікатор користувача	integer	X	X	X	
UserName	ФІО користувача	Text	X			
UserLogin	логін користувача	Text	X			
UserPass	Пароль користувача	Text	X			
AdminRight	Адміністратор чи ні	Boolean	X			
WhenDo	Історія авторизацій та дій	Text	X			

До складу таблиці 5.2 «CNC_Machines» входять наступні поля:

- назва станка;
- специфікація;
- операції що доступні на станку;
- матеріали що станок витрачає під час роботи;
- програми що здатні працювати з цим станком;
- матеріали що можуть бути оброблені на станку;
- витрати енергії станком.

Таблиця 5.2 – «CNC_Machines»

Код	Опис	Тип даних	Обов’язкове	Унікальне	Первинний ключ	Зовнішній ключ
Id	Ідентифікатор станка	integer	X	X	X	
CNCName	Назва моделі станка	Text	X			
Info	Коротко про особливості	Text	X			
CanDo	Операції що станок може робити	Text	X			
Spend	Витрати свердл, змазки…	double	X			
Program	Програми що підтримує станок	Text	X			
Material	Матеріали що станок може обробляти	Text	X			
Energy	Енергія що станок витрачає під час роботи	double	X			

В даній таблиці зібрана уся основні інформація про кожен станок, що може бути використана під час роботи, та при плануванні розкладу.

До складу таблиці 5.3 «CNC_Machines_Parks» входять наступні поля:

- назва станка;
- кількість таких станків;
- продуктивність станків;
- доступність;

Таблиця 5.3 – «CNC_Machines_Parks»

Код	Опис	Тип даних	Обов'язкове	Унікальне	Первинний ключ	Зовнішній ключ
Id	Ідентифікатор групи станків	integer	X	X	X	
CNCName	Назва моделі станка	Text	X			
Number	Кількість станків даного типу	integer	X			
Working	Швидкість виконання операцій	Text	X			
CanUse	Чи доступний станок в даний час	boolean	X			

Дана таблиця відповідає за роботу з групами станків які можна використовувати в даний час.

До складу таблиці 5.4 «Jobs» входять наступні поля:

- назва деталі;
- опис деталі;
- матеріал;
- ціна;
- ціна заготовки;
- модель деталі;

Таблиця 5.4 – «Jobs»

Код	Опис	Тип даних	Обов’язкове	Унікальне	Первинний ключ	Зовнішній ключ
Id	Ідентифікатор деталі	integer	X	X	X	
Name	Назва деталі	Text	X			
Desc	Опис деталі	Text	X			
Material	Матеріал деталі	Text	X			
Price	Ціна готової деталі	double	X			
Price2	Ціна заготовки	double	X			
Model	Модель деталі	Text	X			

В цій таблиці наведено опис деталей, та зібрана уся інформація про них. До складу таблиці 5.5 «Orders» входять наступні поля:

- назва деталі;
- номер замовлення;
- кількість деталей;

- дата замовлення;
- дата дедлайу;

Таблиця 5.5 – «Orders»

Код	Опис	Тип даних	Обов'язкове	Унікальне	Первинний ключ	Зовнішній ключ
Id	Ідентифікатор деталі	integer	X	X	X	
Name	Назва деталі	Text	X			
Number	Опис деталі	integer	X			
NumberD	Матеріал деталі	integer	X			
Date	Ціна готової деталі	date	X			
Dedline	Ціна заготовки	date	X			

В даній таблиці зібрана уся інформація про замовлення що є в системі, кількість деталей що треба виготовити, строки, та відповідно номери замовлень для того щоб відрізняти партії.

До складу таблиці 5.6 «Timetables» входять наступні поля:

- назва деталі;
- назва станка;
- час початку роботи над деталлю;
- час завершення;
- виконання;
- розклад;

Таблиця 5.6 – «Timetables»

Код	Опис	Тип даних	Обов'язкове	Унікальне	Первинний ключ	Зовнішній ключ
Id	Ідентифікатор розкладу	integer	X	X	X	
NameD	Назва деталі	Text	X			
NameCNC	Назва станка	Text	X			
TStart	Час початку	date	X			
TEnd	Час завершення	date	X			
Done	Виготовлено	bulean	X			
NameTT	Розклад	Text	X			

В даній таблиці зберігається інформація про розклади роботи над замовленнями з попередньої таблиці, та розподіл конкретних деталей по конкретним машинам.

До складу таблиці 5.7 «Reports» входять наступні поля:

- назва звіту;
- виготовлено;
- витрачено;

Таблиця 5.7 – «Reports»

Код	Опис	Тип даних	Обов'язкове	Унікальне	Первинний ключ	Зовнішній ключ
Id	Ідентифікатор звіту	integer	X	X	X	
Name	Назва звіту	Text	X			
Earn	В текстовому форматі збережено звіт по доходам з партії	Text	X			
Spend	В текстовому форматі збережено звіт з витратами	Text	X			

В даній таблиці зберігаються звіти по завершеним розкладам.

До складу таблиці 5.8 «Prices» входять наступні поля:

- назва трати (свердло, запчастина, тощо);
- ціна покупки;

Таблиця 5.8 – «Prices»

Код	Опис	Тип даних	Обов’язкове	Унікальне	Первинний ключ	Зовнішній ключ
Id	Ідентифікатор Трати	integer	X	X	X	
Name	Назва Трати	Text	X			
Price	Скільки коштує	double	X			

В даній таблиці зберігаються ціни так званих розхідних матеріалів, в тому числі й електроенергія.

До складу таблиці 5.9 «Programs» входять наступні поля:

- назва програми;
- з якими форматами працює;
- підтримується бліотекою чи ні;

Таблиця 5.9 – «Programs»

Код	Опис	Тип даних	Обов’язкове	Унікальне	Первинний ключ	Зовнішній ключ
Id	Ідентифікатор програми	integer	X	X	X	
Name	Назва програми	Text	X			

Продовження таблиці 5.9

Formats	Формати що обробляє програма, та доступ по API	Text	X			
CanWork	Чи є бібліотека для даної програми в системі	boolean	X			

Таблиця відповідає за роботу з програмами безпосереднього управління станками з ЧПУ. Даний розділ став основою сформованої по даним в ньому ER-діаграми що наведена в додатку 8.

Висновки до розділу 5

В розділі було описано атрибути сутностей системи. Описано основні таблиці бази даних, також на основі наведених вище таблиць, в самій програмі було створено відповідні класи, що й представляють об'єкти збережені в базі даних. Відповідно ці об'єкти використовуються в ході роботи самої системи.

Список наведених вище сутностей, з атрибутами що їм відповідають достатньо для задоволення вимог що було висунуто до системи. При цьому кожна таблиця обов’язкова, і не можемо сказати що якась сутність зайва, оскільки це був би негативний фактор.

Також деякі поля таблиць, зберігають великі об’єми інформації.

Визначившись зі структурою бази даних в системі, можливо починати реалізацію самої системи.

6. РЕАЛІЗАЦІЯ ЛОГІКИ СИСТЕМИ

Після того як було обрано набір технологій, що будуть використовуватись для роботи системи, та після того як було проаналізовано усі виставлені до системи вимоги, було вирішено, що система матиме певну модульну структуру.

Тобто система складатиметься з кількох основних блоків, програма що відповідає за оновлення системи, програма що взаємодіє з користувачем, та набір бібліотек класів, написаних на мові програмуваня C# з використанням платформи .NET 5.0.

Окремо слід зазначити наступні етапи в розробці програмної частини системи:

- створення інтерфейсу користувача;
- реалізація системи реєстрації;
- реалізація системи авторизації;
- реалізація системи зчитування внесених користувачем даних;
- реалізація системи зчитування даних з файлів;
- створення бази даних для системи;
- створення об’єктів та класів, що відповідатимуть таблицям бази даних через Entity Framework Core 5.0;
- реалізація системи взаємодії з базою даних;
- реалізація евристичного алгоритму створення розкладу, на основі даних внесених користувачем до БД;
- реалізація алгоритму методу гілок та меж;
- реалізація процесу формування звіту;
- реалізація процесу формування розкладу;
- реалізація системи моніторингу перебігу роботи;
- реалізація системи виведення файлу звіту;
- реалізація системи управління групою станків з ЧПУ;
- реалізація зовнішніх бібліотек класів, що відповідатимуть за взаємодію з програмами безпосереднього управляння станками з числовим програмним управлінням.

Розробка системи в такому порядку й по таким крокам дозволить створити робочу систему за мінімальний час та без надмірних зусиль, при цьому повність задовольнивши усі вимоги, що були висунуті до системи на попередніх етапах розробки.

6.1. Системи авторизації та реєстрації

Ці дві підсистеми відповідають за безпеку доступу до даних в системі, та безпеку роботи з системою (захищають від спроб отримати конфіденційну інформацію, від спроб несанкціонованого використання системи сторонніми людьми, тощо). Повна версія діаграми в додатку 4.

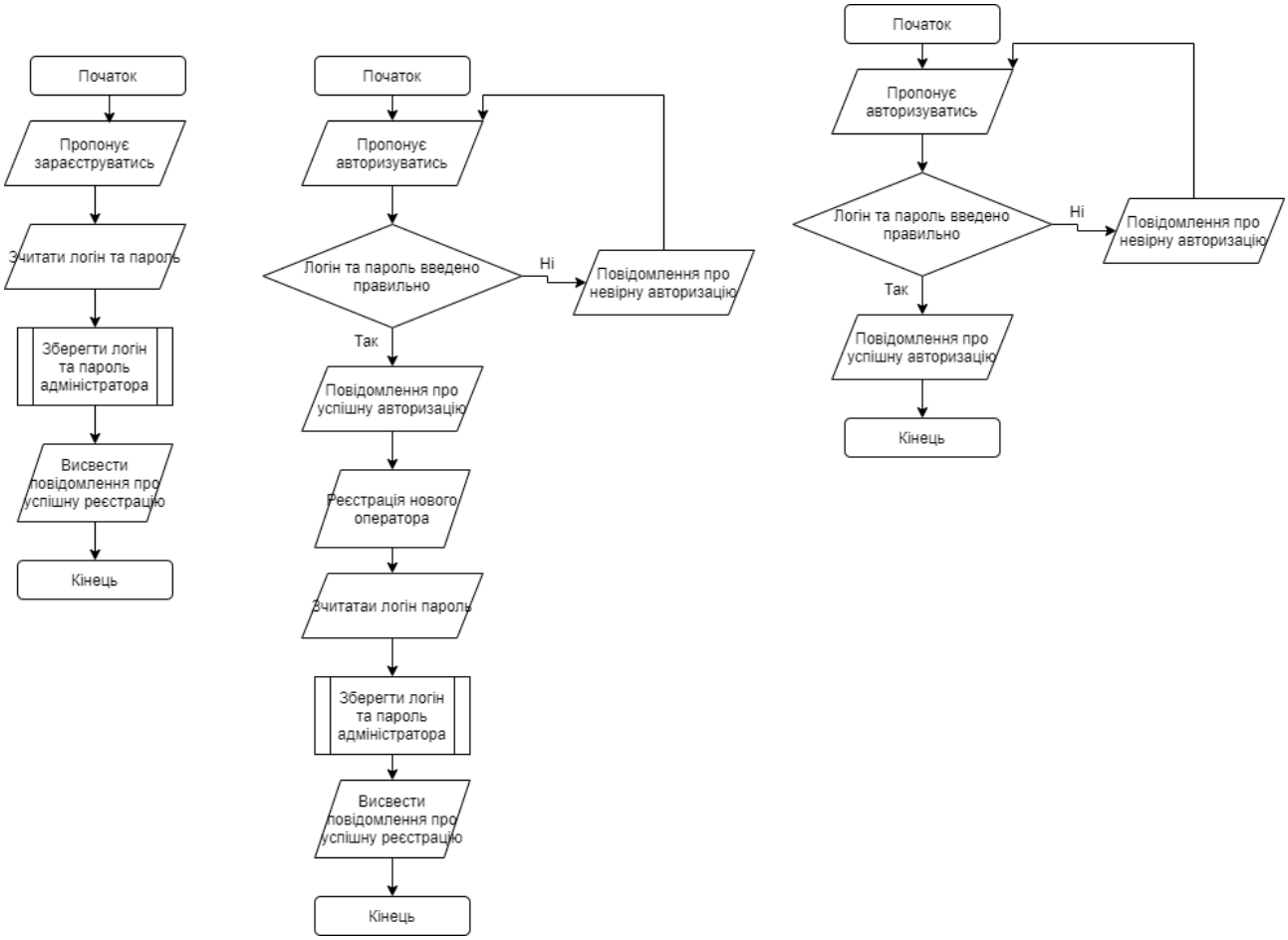

Рисунок 6.1 – Блок-схема модулів реєстрації адміністратора, реєстрації оператора, авторизації оператора

Ці дві підсистеми, що наведені на рисунку 6.1 є дуже важливими в будь якій програмі, адже без них неможливо відслідковувати, хто взагалі працює з програмою, відповідно в системі є два типи акторів, адміністратор – має повний доступ до усього функціоналу системи, та оператори – їх можу бути кілька. Адміністратор реєструється першим, при запуску системи перший раз, потім він має зареєструвати опрераторів.

Авторизіція відбувається наступним шляхом: користувач вводить свій логін та пароль, після чого в залежності від того які в оператора права він або отримує повний доступ до функціоналу системи або частковий. Детальніше про це можна подивитися в додатку 4.

6.2. Система зчитування даних з файлу

Система що відповідає за зчитування даних з зовнішнього файлу та їх запис до бази даних у системі. Це дуже важлива задача, адже вона дозволяє значно спростити введення інформації до системи, так як користувачу не потрібно вручну вводити всю інформацію, він лише вводить певний файл, а система вже сама бере з цього файлу усі потрібні їй для роботи дані, та зберігає їх.

Система ускладнюється тим що для роботи вона повинна вміти знаходити в файлі усю потрібну їй інформацію, також система має розрізняти яку інформацію взагалі записано в файлі (інформація про замовлення, інформація про модель фігури, інформація про станок, файл з незрозумілим системі текстом або форматом). Блок схема роботи цієї системи наведена на рисунку 6.2.

6.3. Реалізація алгоритмів в системі

Теорія розкладів, являється окремим розділом дискрет-ної математики, і відповідає за вирішення проблем впо-рядкування задач.

Рисунок 6.2 – Блок-схема модуля зчитування інформації з файлу

В загальному випадку розглядають дві множини: перша множина, множина машин (приладів, а в нашому ви-падку станків з ЧПУ), друга множина, набір робіт (окремих завдань, або пакетів задач, що потрібно виконати для досягнення поставлених цілей, в нашому випаду виготовлення готової деталі з заготовки). Перед нами стоїть задача дискретної оптимізації, побудувати розклад який буде мінімізувати час витрачений на виконання усього списку робіт. Розклад, це набір інструкцій, в яких вказано на яких машинах, і в якому порядку слід виконувати роботи.

Нас цікавить група задач теорії розкладів без переривань, оскільки навіть при конвеєрному виробництві, на одному станку кожна робота (навіть якщо це лише части-на виробничого циклу) виконується від початку до свого завершення без переривань.

Для нашої задачі складання розкладу роботи групи станків слід зазначити, що в кожний момент часу кожна машина виконує не більше однієї роботи (один станок не може одночасно виконувати дві програми, навіть коли з однієї заготовки створюємо декілька деталей, він виконує один набір команд, а отже це являється однією робо-тою), і кожна робота виконується на одній машині, або не виконується взагалі.

Зокрема нас цікавить клас задач з директивними строками, так як для виготовлення кожної деталі задано час виготовлення, та директивний строк завершення роботи над деталлю. Тут нам потрібно вирішити два завдання:

1) знайти такий розклад, що буде задовольняти директивні строки;

2) якщо такий розклад існує, треба розв’язати задачу мінімізації часу простою;

Для розв’язання поставленої задачі були використані такі алгоритми [5]:

1) евристичні алгоритми – алгоритми засновані на певних математичних припущеннях про характеристики оптимального рішення задачі. Але так як цей алгоритм може дати будь-яку похибку відносно оптимального рішення, було використано його більш просунуту версію «Алгоритм ймовірнісного пошуку»;

2) метод «Гілок і Меж» або (branch and bound) –являє собою розвиток методу повного перебору, але на відміну від нього, він відкидає ті рішення які не можуть мати в собі оптимальне рішення. Застосовується для знаходження еталонного рішення;

Детальніше про алгоритм методу «Гілок і Меж»: розбиваємо основну множину задач, на підмножини (підзадачі), цей процес відбувається рекурсивно, тобто підзадачі в свою чергу розділяються на менші підзадачі. В ході розгалудження отримуємо дерево пошуку оптимального рішення. В ході роботи алгоритму будується кілька допустимих рішень, в пам’яті зберігаємо найкраще, його називають

«Рекордом». І якщо нижня оцінка окремої гілки більша рекорду, її відсікаємо. Алгоритми наведені в додатку 5.

Алгоритм роботи програми:

1) за допомогою евристичного алгоритму знаходимо будь-яке допустиме рішення;
2) зберігаємо рішення в пам’яті як рекорд;
3) застосовуємо алгоритм гілок та меж;
4) розбиваємо множину рішень задач на підзадачі;
5) для кожної підзадачі знаходимо нижню оцінку;
6) якщо нижня оцінка більша рекорду, відкидаємо гілку;
7) інакше, якщо в гілці лише одне рішення, приймаємо нижню оцінку за новий рекорд, якщо в гілці кілька рішень, знову застосовуємо алгоритм з 5 кроку;

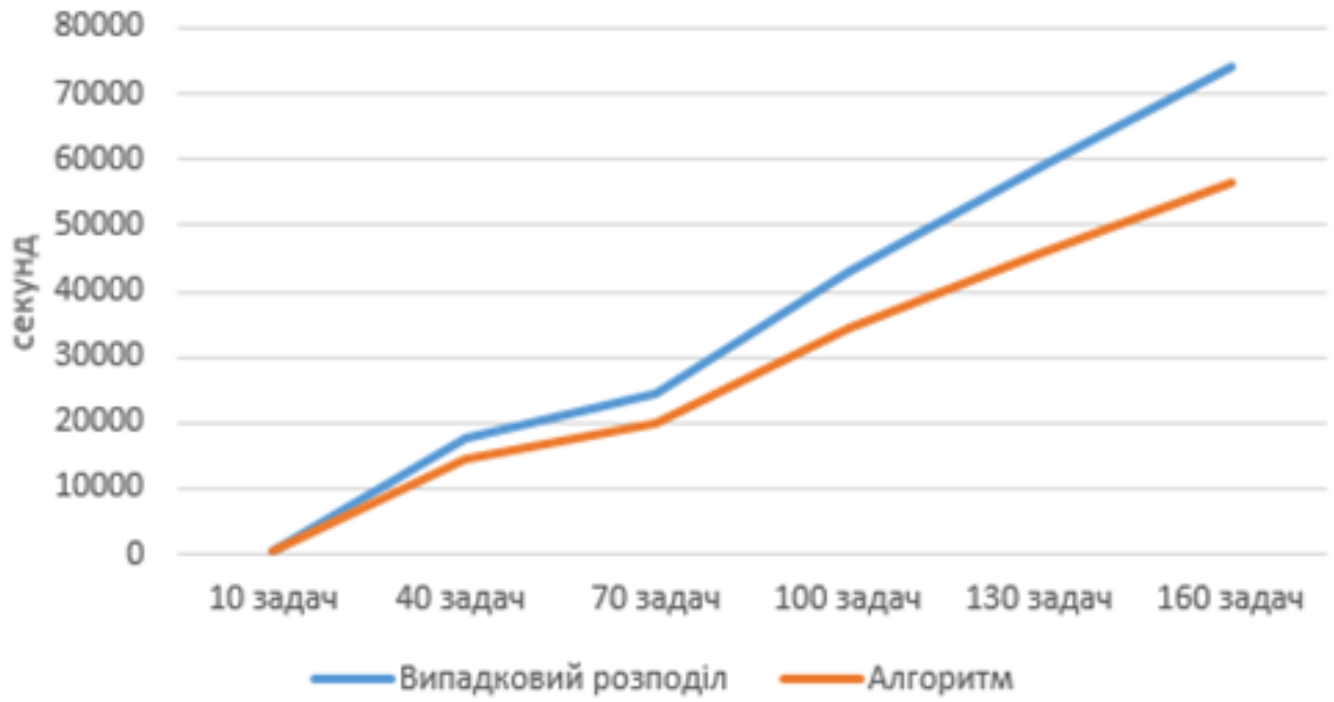

Рисунок 6.3 – Графік порівняння часу при різних алгоритмах

Не дивлячись на те, що такий алгоритм дуже ресурсозатратний його використання дозволяє значно підвищити ефективність роботи групи станків. При чому чим склад-ніше замовлення, тим вища ефективність. На рисунку 6.3 наведено

графік роботи групи з 4 станків над різною кіль-кістю задач. Верхній графік, задачі розподілялись в випадковому порядку (що приводило до порушення директив-них строків), нижній графік розподіл задач здійснювався за алгоритмом. З зростанням кількості задач різниця в часі на їх виконання зростає, і при 160 задачах, при розкладі створеному алгоритмічно ми отримуємо виграш в 24%, або 4 години 57 хвилин. Для перевірки боло взято два однакових набори задач, та одні й ті самі станки, відрізнялися лише підходи до складання розкладу роботи. При складанні розкладу випадково не було враховано такі особливості як швидкість обробки певних типів деталей на певних станках, так ніяк не бралося до уваги, що при зміні типу деталей, станку потрібен час на переналаштування.

6.4. Система управління станком з ЧПУ

Наша система на пряму не керує станками з числовим програмним управлінням, це було встановлено та обгрунтовано ще в перших розділах магістерської дисертації. Відповідно за роботу з станками відповідають програми управління станками з ЧПУ. Система ж лише моніторить стан кожного станка з ЧПУ, та відповідно віддає команди програмі безпосереднього управління станком з ЧПУ, деталі того як це відбувається можна побачити на рисунку 6.4.

Дана блок-схема зображує роботу лише з одним станком, програма ж підтримує до тридцяти станків одночасно, відповідно робота з кожним зі станків відбувається асинхронно, тобто кожен станок оброблюється в окремому потоці, операції ж запису інформації до бази даних реалізовані за допомогою такої функції мови програмування C# як локери, тобто доступ до конкретного сегменту коду може мати лише один потік в одиницю часу, інші потоки очікують завершення головного.

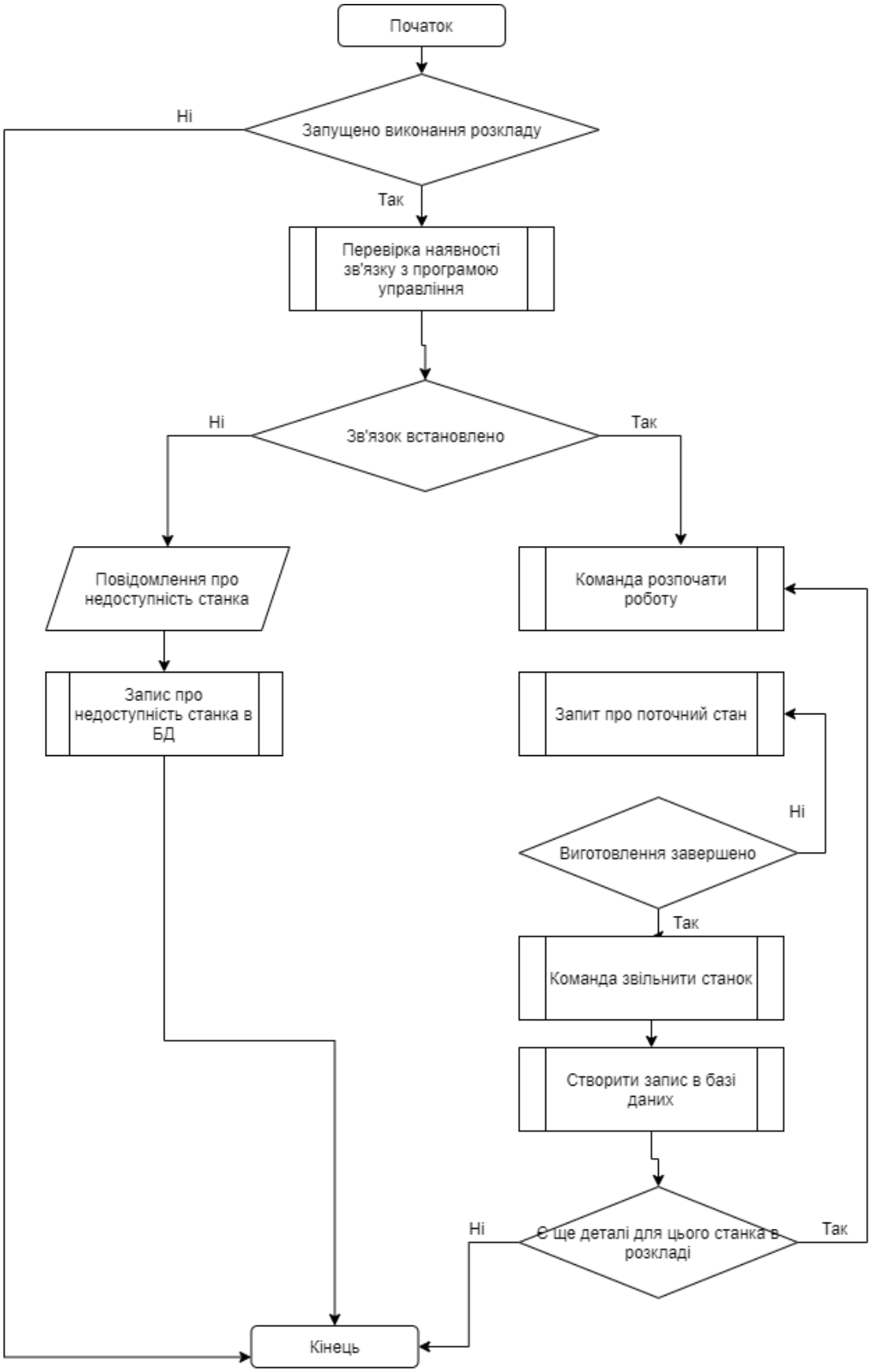

Рисунок 6.4 – Блок-схема зв’язку з програмою управління по API

Висновки до розділу 6

Так як система має різні модулі з яких вона власне і складається, процес написання коду може бути розділеним на різні етапи, наприклад інтерфейс користувача було реалізовано вже після того як були написані блоки що відповідають за генерацію розкладу, звіту, та роботи з базою даних. При цьому інтерфейс користувача вдалося підключити до системи дуже просто й швидко.

В цьому розділі наведено реалізацію не всіх модулів системи, а лише тих що виглядали цікаво. Було розглянуто кілька ключових модулів, без яких система не могла б працювати в принципі.

Після того як було завершення реалізації усіх модулів системи, залишається лише завершити реалізацію інтерфейсу користувача, та протестувати готову систему на предмет наявності помилок.

7. РОЗРОБЛЕННЯ ІНТЕРФЕЙСА КОРИСТУВАЧА

Інтерфейс користувача, один з ключових елементів будь якої програми, адже без нього неможливо уявити що програма отримає бодай якусь широку популярність. В сучасному світі користувач не буде працювати з системою через консоль. Також в консолі неможливо візуалізувати різні дані у вигляді таблиць або графіків, все це можливо лише з використанням графічного інтерфейсу.

В попередніх розділах було вирішено робити графічний інтерфейс нашого додатку з допомогою такої технології як WPF або Windows Presentation на платформі Foundation .NET 5.0.

Нижче представлені основні модулі системи, а точніше наведено їх графічне відображення на рисунках 7.1-7.4.

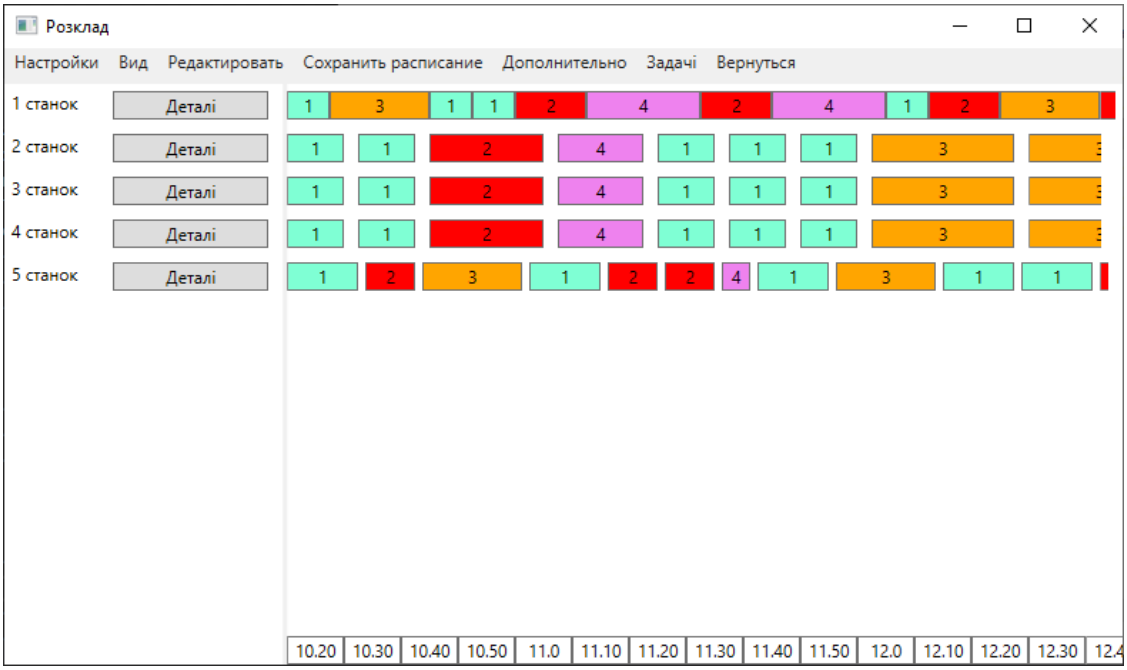

Рисунок 7.1 – Інтерфейс управління розкладом

На рисунку зображено розклад для групи з п’яти станків, при чому три станки однієї моделі, тому час виконання для них ідентичний, як і набори задач, загалом задача стояла в виготовленні партії деталей чотирьох видів, та на рисунку наведено бачимо розподіл задач по кожному станку.

Внизу вікна знаходиться часова шкала, в будь який момент часу користувач може дізнатись всю наявну інформацію про будь який станок або про будь яку деталь.

Групи деталей відрізняються між собою кольорами, це зроблено для того, щоб було візуально простіше відрізняти між собою групи різних деталей.

В кожного станка є свій набір задач, що він повинен виконати, візуально вони згруповані в лінію навпроти номеру станка. Також поряд з кожним номером станка є кнопка «Деталі». Натиснувши її можна переглянути специфікацію кожного станка.

Специфікація кожного станка відкривається в окремому вікні, що зображено на рисунку 7.2. Там виводиться інформація з бази даних про кожен конкретний станок. Відповідно там також буде зображення станка, якщо його було завантажено до бази даних раніше.

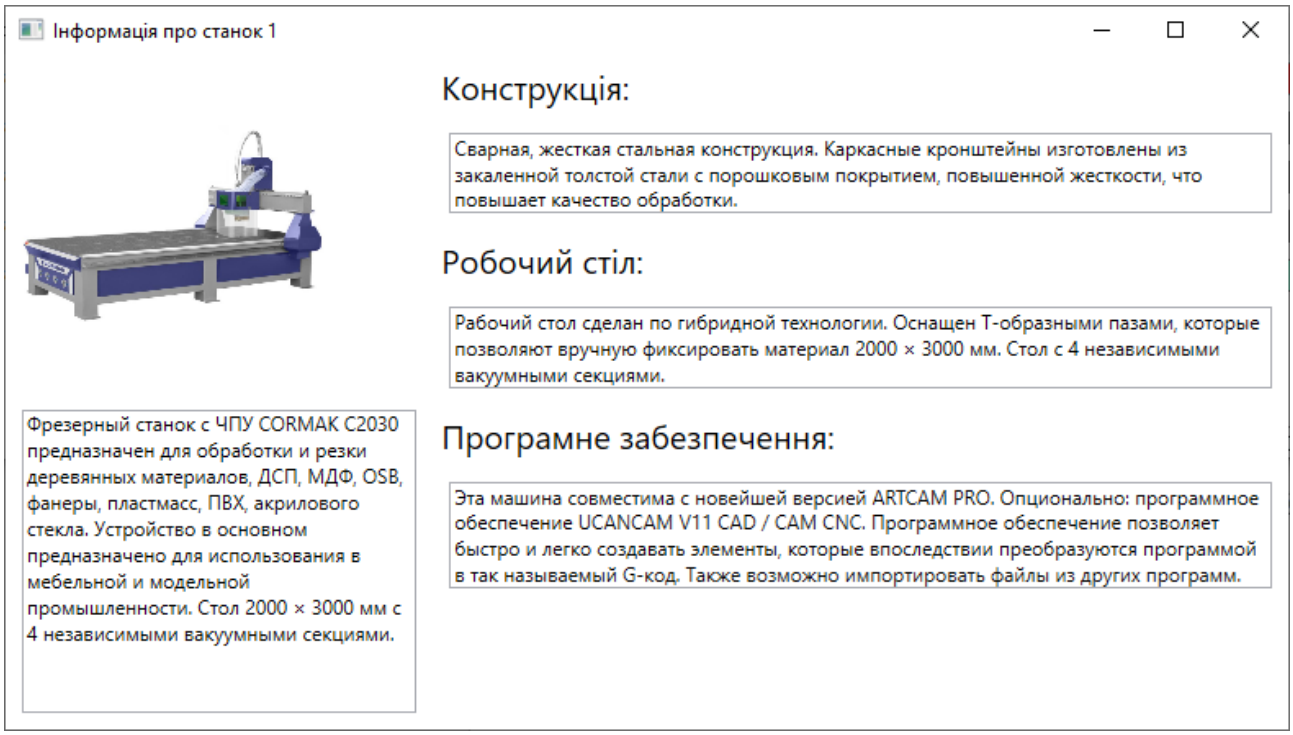

Рисунок 7.2 – Інтерфейс вікна інформації про станок

Під зображенням станку виведено його назву та короткі відомості. В полі «Конструкція» Виведено інформацію про конструктивні особливості станку, що можуть бути важливими при роботі зі станком.

В полі «Робочий стіл» виведено відомості про робочу поверхню станка, ця інформація дозволяє зрозуміти які габарити взагалі повинні бути у заготовок та

кінцевих деталей для того щоб програма ефективно працювала, та станок не ламався під час експлуатації. Також тут зібрана інформація про те наскільки автономний взагалі станок, потрібне втручання оператора в роботу чи ні, якщо потрібне то коли і в якій мірі.

Поле «програмне забезпечення» включає в себе інформацію про програмне забезпечення станка, ця інформація важлива як системі (оскільки на її основі виконується вибір бібліотек до відповідних програм управління, та вибір самих програм управління станком з числовим програмним управлінням.

Важливість цієї інформації важко переоцінити, оскільки саме на її основі можна сказати й працює безпосередньо система, так як основне її завдання для кожного станка використовувати саме ту програму управління, що максимально ефективно розкриє можливості кожного конкретного станка.

Усі завдання, що було зображено на рисунку 7.1 відрізняються між собою не лише кольорами, а й числами. Це номери моделей деталей. Саме вони використовуються системою для внутрішньої роботи, кольори це лише певний зовнішній прояв, що полегшує роботу користувача, адже розділити групи деталей по кольорам простіше, ніж розділити деталі по номерам, оскільки різні кольори сприймаються оком легше ніж різні числа на одному й тому самому фоні.

Тим не менше, усі ці різнокольорові сегменти являються кнопками, тобто якщо користувач не в режимі редагування натисне на будь яку деталь, він отримає повну інформацію про неї в окремому вікні. Наприклад на рисунку 7.3 зображено вікно з інформацією про деталь під номером 3.

Інформація про кожну деталь виводиться в окремому вікні, основний інтерфейс розкладу не закривається, при наведенні на кожну кнопку спливає підказка з назвою деталі. Якщо ж натиснути на кнопку, замість підказки відкривається окреме вікно, що бачимо на рисунку 7.3.

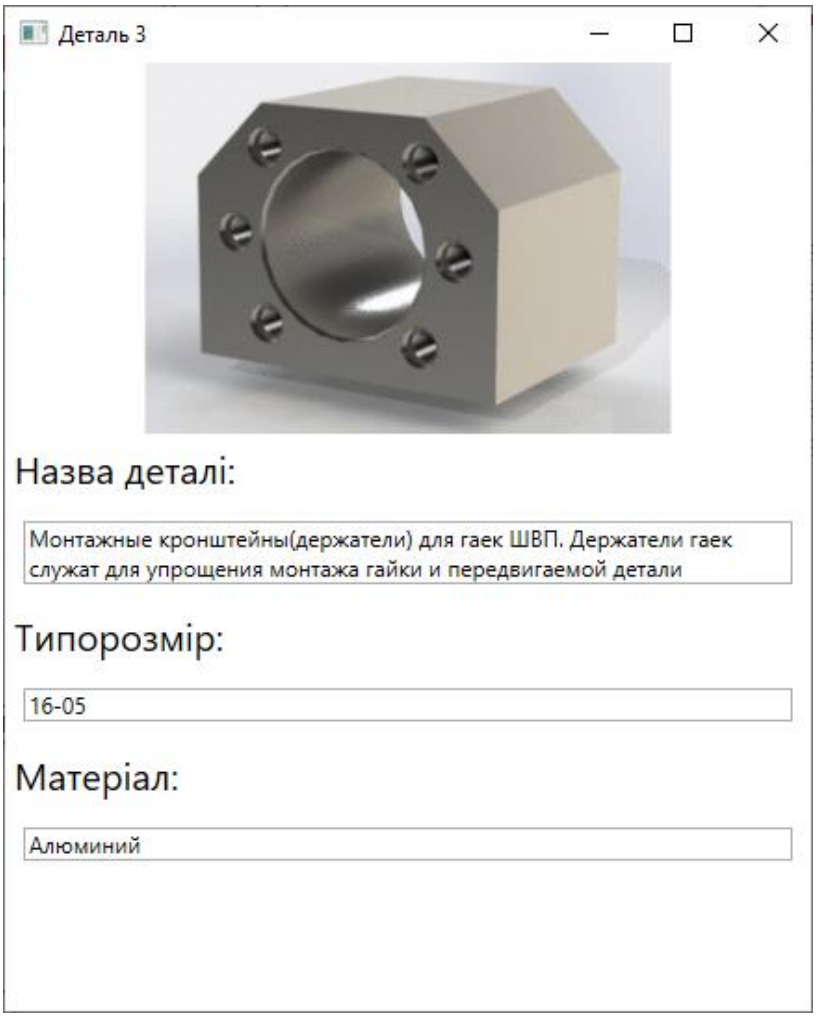

Рисунок 7.3 – Інтерфейс вікна інформації про деталь

Тут в нас виводиться більш детальна інформація про деталь, а саме можемо побачити:

- зображення. (виводиться або конкретне зображення що було завантажено до бази даних, або рендер моделі готової деталі, що також був завантжений заздалегідь;
- в полі «Назва деталі» виведено інформацію про назву готової деталі, повна назва або скорочена, можливо якісь конструктивні особливості деталі;
- в полі «Типорозмір» виводиться коротка інформація про розміри деталі, відповідно це поле пов’язане з попереднім, адже типорозміри деталей різних груп, наприклад кулька металу, та болт матимуть різні типорозміри, а наприклад гайки різного розміру належать до однієї групи деталей, та їх типорозміри мають пряму залежність, тож це поле важливе;

- в полі матеріал в нас виводиться інформація про той матеріал з якого має бути заготовка та кінцева деталь, це дуже важлива інформація, так як на її основі виконується прийняття рішення що до того чи можливо виготовляти конкретну деталь на конкретному станку, або ж, якщо це важливо то відбувається зміна оброблюючої оснастки на стаку (свердла, фрези, тощо);

Далі зображено вивід розкладу в системі. Це лише попередній його вигляд, тобто тут в нас виведено лише коротку інформацію про те яку саме партію деталей було виготовлено. Зображення цього вікна наведено на рисунку під номером 7.4

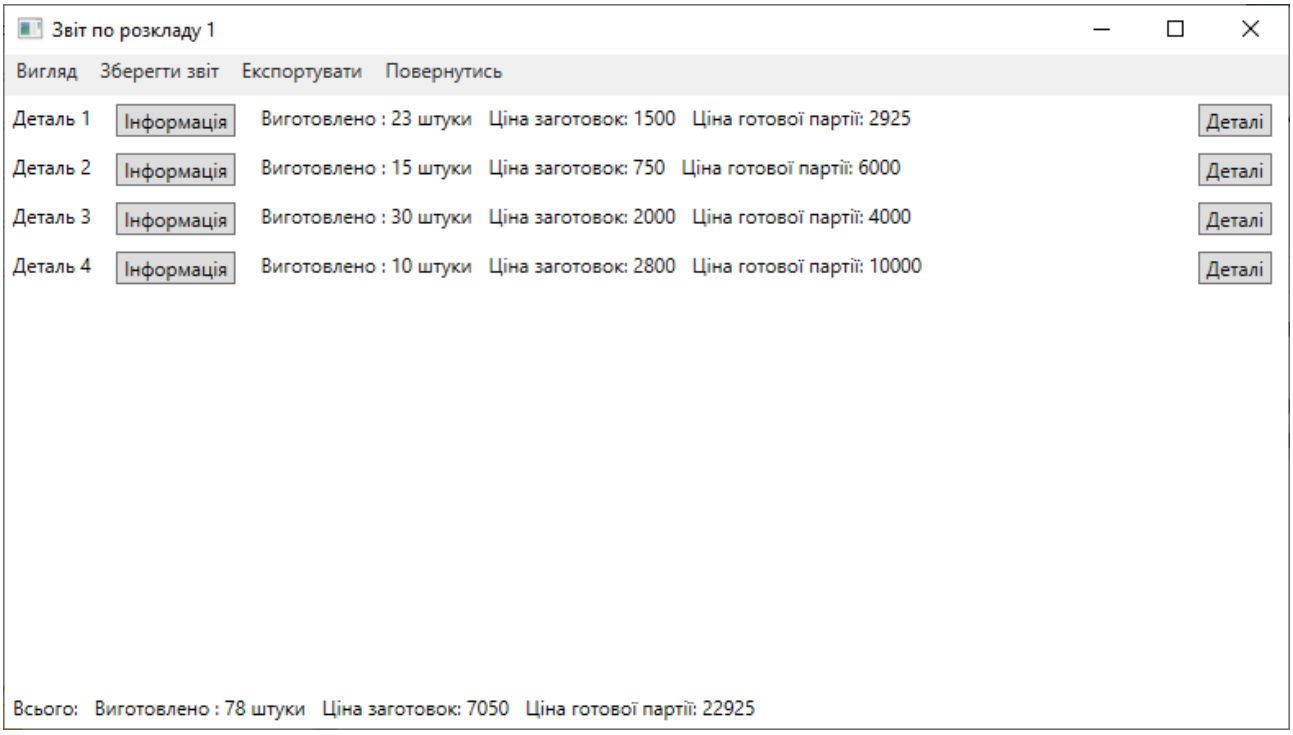

Рисунок 7.4 – Інтерфейс вікна звіту

Відповідно на вікні звіту про виконання розкладу в нас виведено попередню інформацію про виконання розкладу. В нас на екран виводиться список усіх видів деталей що виготовлялись під час відповідного розкладу.

Тут першим іде назва деталі (точніше не назва самої деталі, а назва її типу), а точніше її номер. Далі йде кнопка «Інформація», натискаючи її відкривається інформація про деталі, що були наведені на рисунку 7.3. Це повністю аналогічні вікна (по факту вони являються одним і тим самим вікном). Далі йде інформація про партію

деталей, а саме йде кількість виготовлених деталей, ціна закупки партії заготовок нових деталей, та далі йде ціна вже виготовленої партії готових деталей. Справа вікна знаходиться кнопка «Деталі». Тут знаходиться більш розгорнутий звіт, з врахуванням ціни амортизації станків, ціна на затрачену електроенергію, часові рамки виготовлення кожної окремої деталі, інформація про збої під час роботи (якщо вони були).

Внизу вікна що наведено на рисунку 7.4. наведено звіт по всьому розкладу, там підсумовано усю інформацію по всім видам деталей.

Висновки до розділу 7

В даному розділі наведено основні елементи інтерфейсу користувачів, що відображають основний функціонал системи. Тут можна побачити те як виглядає інтерфейс користувача. Тож не дивно що в цьому розділі зображено основну кількість зображень майбутнього вигляду системи. На даному етапі система вже повністю в робочому стані, та готова до використання.

8. РОЗРОБКА СТАРТАП-ПРОЕКТУ

Стартап проект – це певна ідея, що є унікальною та вона покликана вирішити проблеми, що наявні в певній сфері, також проект має принести певний дохід, фінансовий, ресурсний чи можливо визначений іншими величинами. Зазвичай стартап це якісь нові, інноваційні ідеї чи рішення, що раніше не був представлений на ринку.

Однак доволі рідко можливо створити щось абсолютно нове, чому взагалі немає аналогів, тож стартап це може бути якийсь новий підхид, ідея що покращить або полегшить виконання якогось уже використовуючогося алгоритму.

8.1. Опис ідеї проекту

Проект: це система керування групою станків з ЧПУ. Особливості системи в тому, що розклад роботи для групи станків з ЧПУ буде генеруватись на основі замовлення. В таблиці 8.1 описано ідеї стартап-проекту.

Основні технічні та економічні особливості:

- збереження та обробка інформації;
- генерація розкладу;
- керування групою станків з ЧПУ;
- збір економічної інформації;
- налаштування системи під користувача;
- перегляд перебігу виконання замовлення;
- створення звіту з результатами роботи;
- можливість працювати з різними ПУ;
- можливість працювати з різними видами станків з ЧПУ;
- можливість підключати зовнішні бібліотеки;
- можливість отримувати структуровану інформацію з файлів.

Таблиця 8.1 – Ідея стартап-проекту

Зміст ідеї	Напрямки використання	Вигоди для користувача
	1. Підвищення ефективності роботи групи станків з ЧПУ	Алгоритми теорії розкладів дозволять значно підвищити ефективність роботи групи чпу станків, оскільки задачі будуть розподілятись відповідно можливостей кожного станка, також це в свою чергу дозволить значно підвищити ефективність шляхом зменшення або нівілювання часу простою кожного конкретного станка
	2. Збір інформації про витрачені ресурси, та шляхи оптимізації, що підійдуть для роботи з групою станків з ЧПУ.	Система автоматично підраховує усі витрати що можуть виникати під час роботи, будь то заготовки, електроенергія, чи різні матеріали та речовини що витрачає станок під час своєє роботи, свердла, мастило, тощо. Це дозволить швидко аналізувати економічні вигоди.

В таблиці 8.2 наведені сильні і слабкі сторони в порівнянні з конкурентами.

Таблиця 8.2 –Слабкі сильні та нейтральні сторони проекту

№	Характеристики	Товари конкурентів				W	N	S
		Мій	LinuxCNC	ArtCAM Pro	CLOBBI			
1	Зберігання моделі деталі	+	+	+	+	-	+	-
2	Зберігання інформації про станок	+	-	-	-	-	-	+
3	Створення розкладу роботи	+	-	-	+	-	-	+
4	Автометичне виготовлення групи деталец	+	-	-	-	-	-	+
5	Створення економічного звіту про результати роботи	+	-	-	+	-	-	+
6	Можливість переглядати поточний стан виготовлення замовлення	+	-	-	-	-	-	+

8.2. Технологічний аудит

На цьому етапі було проаналізовано доступність та наявність тих рішень, що будуть використовуватись у проекті, та приведено їх опис і аналіз у таблиці 8.3.

Таблиця 8.3 – Технологічна здійсненність ідеї проекту.

№	Ідея проекту	Технології її реалізації	Наявність технологій	Доступність технологій
1	Збереження даних користува, та можливість їх подальшого використання	MS SQL server	Наявна	Доступна
2	Провайдер, що реалізовує доступ до самої бази даних, з коду програми та дозволяє програмі працювати	EF Core 5.0	Наявна	Доступна
3	Можливість створити інтерфейс десктопного додатку, та використовувати його	WPF	Наявна	Доступна
4	Можливість реалізувати алгоритми теорії розкладів на практиці та використовувати їх	Мова програмування C#	Наявна	Доступна
5	Платформа що дозволить реалізувати модульну структуру програми	.NET 5.0	Наявна	Доступна
Обрана технологія реалізації ідеї проекту: є можливою.				

Усі перераховані вище технології легкодоступні, та прості в своєму використанні, тим не менше деякі з перерахованих технологій є платними, але не дорогими.

8.3. Аналіз ринкових можливостей запуску стартап-проекту

Ринок ЧПУ станків являється основою сучасної проми-словості, оскільки лише він здатен забезпечити необхідний рівень швидкості й точності обробки деталей, при високих показниках продуктивності та гнучкості виробництва (легко змінювати тип деталей що виготовляються). Так як технології 3D друку, незважаючи на значний прогрес, все ще залишаються вузькоспеціалізованими і не можуть кон-курувати з традиційними обробляючими центрами, особ-ливо в сфері металообробки. За даними інтернет видавни-цтва МНИАП світовий ринок станків з ЧПУ виросте з ни-нішнього $ 93 млрд до $ 128 млрд до 2026р. Очікується, що в найближче десятиліття основним напрямком розвит-ку стануть ЧПУ-станки, які використовуються в автомобі-льній промисловості, зокрема станки що використовують-ся для формування корпусу автомобіля. Інший перспекти-вний напрямок розвитку, це точне машинобудування. Використовується в авіакосмічній та оборонній промис-ловості, при виготовленні медичного обладнання та техні-ки. Складність виробів, використання особливих металів та сплавів, роблять переваги використання станків з ЧПУ вирішальними для широкого їх застосування в наведених вище сегментах ринку [5].

Оскільки ринок зростає, можна сказати що він є цікавим для інвестицій та нових проектів. Далі у таблиці 8.4 наведено характеристики клієнтів. Там же розглянуто питання щодо цільової аудиторії, «клімат», що було сформовано на ринку, описано можливі відмінності між різними групами цільової аудиторії. Зокрема в цій же таблиці виписано основні вимоги споживачів що було поставлено перед програмним продуктом.

Таблиця 8.4 – Характеристика потенційних клієнтів стартап-проекту

Потреба, що формує ринок	Цільова аудиторія	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару
Потребує підвищення ефективності планування виготовлення партій товарів, зменшення часу простою станків, зниження витрат що виникають через неефективний розподіл завдань між системами та групами користувачів.	Здебільшого представники малого та середнього бізнесу	Різна поведінка може бути зумовлена різними рівнями підтримки користувачів	-висока надійність -простота -ефективність планування

Далі в таблиці 8.5 розглянуто фактори що загрожують впровадженню продукту на ринок, та фактори що йому сприятимуть.

В таблиці 8.6 описано не лише самі фактори, що можуть становити загрозу, а й детально описано саму загрозу, та варіанти можливих дій що приведуть до ліквідації її та мінімізації збитків.

Таблиця 8.5 – Фактори загроз стартап-проекту

№	Фактор	Зміст загрози	Можлива реакція компанії
1	Залежність від зовнішніх програм управління станками з ЧПУ	Оскільки система сама безпосередньо не керує роботою станків з чпу, можуть виникати помилки та збої в зв’язку з невірно налаштованою програмою управління, що може призводити до псування деталей, або простою	Пошук надійних ПУ
2	Виникнення нових видів станків	На новому станку можуть працювати програми які не додані до системи, при цьому ефективність нових станків з зрозумілих причин буде значно вища з ефективність старих, тож користувач захоче працювати саме на нових	Створення бібліотек під нові ПУ та нові станки

Продовження таблиці 8.5

3	Поява конкурентної компанії	Конкуренти	Детальний аналіз нових гравців на ринку, знаходження слабких та сильних сторін продукту конкуренту
4	Мала кількість продажу товару	Не збулися прогнози продажів	Детальний аналіз усіх доступних відгуків користувачів, пошук шляхів покращення проекту, внесення змін в стартегію маркетингу, внесення змін до рекламної компанії

Таблиця 8.6 – Фактори можливостей стартап-проекту

№	Фактор	Зміст можливості	Можлива реакція компанії
1	Зацікавленість в системі іноземних користувачів	Нові ринки збуту	Аналіз потреб користувачів, доопрацювання системи відповідно до їх потреб, внесення усіх необхідних змін до системи, додавання можливості вибору мови інтерфейсу системи

Продовження таблиці 8.6

2	Заінтересованість в системі інвесторів	Збільшення фінансових можливостей	Масштабування проекту, додавання нових функцій що було вирішено відкласти, розширення команди підтримки проект, підвищення ефективності роботи з проектом

Далі в таблиці 8.7 йде аналіз наявних на ринку конкурентів:

Таблиця 8.7 – Аналіз конкуренції в галузі за М. Портером

Скла-дові аналізу	Прямі конкуренти в галузі	Потецнійні конкуренти	Постачальники	Клієнти	Товари замінники
	Системи LinuxCNC ArtCAM Pro CLOBBI	Конкурен-том може стати будь яка система що буде займатись плануван-ням роботи групи станків, або оптиміза-цією розпорядку	Мають контроль над значною частиною ринку	Малий та середній бізнес	немає

Продовження таблиці 8.7

Вис-новки	Інтенсивність конкурентів доволі висока, конкуренти покривають значну частину ринку, при цьому конкуренти продовжують розвивати свої додатки	Конкуренти можуть з’явитися не відразу	Залежність від постачальників	Система має бути ефективною та надійною	немає

Фактори що підтверджують конкурентоспроможність наведені в таблиці 8.8:

Таблиця 8.8 – Обґрунтування факторів конкурентоспроможності

№	Фактор конкурентоспроможності	Обґрунтування факторів
1	Зручність користування	Продукт значно знижує час користувача що витрачався на планування розкладу роботи над групою товарів, при чому залежно від того яка саме група, наскільки вона велика, та як багато доступних станків з ЧПУ може змінюватись ефективність роботи користувача
2	Створення звітів	Користувачу не потрібно вручну створювати звіти, все що йому треба це натисснути одну кнопку та отримати готовий результат, на основі якого й буде виконано роботу що потрібна для створення звіту по виготовленню партії

Продовження таблиці 8.8

3	Можливість налаштування безпеки	Оскільки в системі є два рівні захисту користувачу не потрібно хвилюватись, що хтось отримає доступ до його статистичної інформації, моделей деталей, або ще якихось важливих елементів що можуть виникати під час роботи системи.

Таблиця 8.9 – Порівняльний аналіз сильних і слабких сторін стартап-проекту

Фактор конкурен-тоспромож-ності	Бали 1-20	Рейтинг конкурентів в порівнянні з стартапом						
		-3	-2	-1	0	1	2	3
Зручність користування	18		ArtCAM Pro	Linux CNC			CLO BBI	
Створення звітів	15	Linux CNC	ArtCAM Pro			CLO BBI		
Можливість налаштуванн я безпеки	15	Linux CNC	ArtCAM Pro		CLOB BI			

Як можна помітити з наведених вище таблиць, то по факторам конкурентноспроможності значні переваги саме на стороні стартап проекту, тому в нього є усі показники для виходу на ринок.

Нижче в таблиці 8.10 наведено SWOT-аналіз стартап-проекту, де прописані сильні та слабкі сторони проекту, а також описані можливості та види загроз, що можуть виникнути.

Таблиця 8.10 – SWOT-аналіз

Сильні сторони: - Актуальність - Унікальні можливості - Підвищення продуктивності роботи групи станків з ЧПУ, шлягом зниженн часу простою, та оптимізації розкладу роботи - Зручність використання системи	Слабкі сторони: - Високий рівень конкуренції - Складно впроваджувати проект в великі системи - Залежність від програм управління станками з ЧПУ
Можливості: - Зацікавленість в системі іноземних користувачів - Заінтересованість в системі інвесторів	Загрози: - Залежність від зовнішніх програм управління станками з ЧПУ - Виникнення нових видів станків - Поява конкурентної компанії - Мала кількість продажу товару

Таблиця 8.11 – Альтернативи ринкового впровадження

Альтернатива (орієнтовний комплекс заходів) ринкової поведінки	Ймовірність отримання ресурсів	Строки реалізації
Загарбник	Середня	Більше 1.5 року
Наступник	Суттєва	1 рік
Виклик лідеру	Низька	Більше 2 років

Для стартапу було визначено три основні альтернативи ринкової поведінки – «наступник», «виклик лідеру» та «загарбник» (деталі таблиця 8.11). Якщо використаємо альтернативу «загарбник» є середня можливість отримати прибуток, так як дана альтернатива залежить від більшої кількості факторів, ніж інші запропоновані. Якщо використовувати альтернативу «наступник» є висока ймовірність отримати прибуток так як буде захоплено конкретну сферу.

Альтернатива «виклик лідеру» майже стовідсотково принесе прибуток, але час на те щоб його отримати буде помітно більший, так як у лідера ринку вже є клієнтська база позитивно налаштована на співпрацю.

8.4. Розроблення ринкової стратегії проекту

Базова стратегія розвитку під час розробки ринкової стратегії наведено в таблиці 8.12.

Таблиця 8.12 – Визначення базової стратегії розвитку

Обрана альтернатива розвитку проекту	Стратегія охоплення ринку	Ключові конкурентоспроможні позиції відповідно до обраної альтернативи	Базова стратегія розвитку
Виклик лідеру	Концентрація на потребах великого сегменту ринку	Надання клієнтам сучасного програмного забезпечення за зниженою ціною	Стратегія диференціації

Вибираючи з існуючих стратегій по розвитку для першої частини життєвого циклу стартапу було обрано стратегію диференціації, так як вона чудово підходить існуючим на ринку умовам, при врахуванні характеристик програми. Наша мета – виділити та відокремити найвагоміші плюси нашого програмного додакту на фоні конкурентів. Далі ми маємо вибрати стратегію нашої поведінки в середовищі конкуренції.

Таблиця 8.13 – Визначення стратегії позиціонування

Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспроможні позиції власного стартаппроекту	Вибір асоціацій, які мають сформувати комплексну позицію власного проекту (три ключових)
Висока ефективність, наявність алгоритму створення розкладу, приємний інтерфейс користувача.	Стратегія диференціації	Стратегія виклику лідера	Ефективність, аналітика, підвищення доходів

Як позиціонувати програму на ринку було вирішено по результатам SWOT-аналізу цільових груп можливих користувачів, було визначено стратегію по підкоренню ринку (таблиця 8.13), та конкурентні переваги стартапу.

8.5. Розроблення маркетингової програми стартап-проекту

Щоб підвищити прибутки від реалізації стартапу було прийнято рішення реалізовувати продукт власними силами, для цього будуть використані інтернет-ресурси для розповсюдження системи, серед низх будуть представлені великі інтернет магазини, такі як магазін Microsoft та інші. А останній крок маркетингової компанії це розробка комунікацій (табл. 8.14).

Таблиця 8.14 – Формування системи збуту

Специфіка закупівельної поведінки цільових клієнтів	Функції збуту, які має виконувати постачальник товару	Глибина каналу збуту	Оптимальна система збуту
Покупка після презентації товару	Доставка, надання консультацій при продажі та використанні	Глибока	Власні сили, платформи для продажу рішень

Маркетингова комунікація це неймовірно важлива складова життєвого циклу стартап проекту, так як від неї в значній мірі залежить успіх проекту в цілому на перших етапах свого розвитку. Далі маркетингова комунікація перейде до наступної фази, замість пошуку нових користувачів вона буде займатись покращенням відношення до проекту уже наявними користувачами, та отримувати притік нових клієнтів на основі відгуків.

У таблиці 8.15 розглядається специфіка поведінки різних цільових груп клієнтів, які потенційно можуть бути у продукту. В цій же таблиці описані канали через які потенційні клієнти обмінюються та сприймають інформацію, що буде корисним для рекламної кампанії. Також тут наведено ключові позиції що було обрано для позиціонування продукту. Описані можливі рекламні повідомлення та звернення, що підвищить впізнаваність товару та торгової марки на ринку, та підвищить прибуток, оскільки чим більше людей знають про компанію, тим більше в неї число потенційних клієнтів.

В сучасному світі основний спосіб комунікації між людьми є Інтернет, тож використання цього каналу обміну інформацією являється обов’язковим. Рекламна компанія в мережі дозволить віщати на максимальну аудиторію потенційних користувачів. Основні ключові особливості стартапу це його ефективність та унікальні можливості.

Таблиця 8.15 – Концепція маркетингових комунікацій

Специфіка поведінки цільових клієнтів	Канали комунікацій, якими користуються цільові клієнти	Ключові позиції, обрані для позиціонування	Завдання рекламного повідомлення	Концепція рекламного звернення
Серед клієнтів багато менеджерів шукають рішення для своїх проблем на спеціалізованих інтернет майданчиках	Інтернет, Сарафанне радіо	Сучасна система дбає, що сприяє ефективному веденню бізнесу	Донести користувачам інформацію про якісний і сучасний продукт з перевагами	Оригінально сповістити споживачів про новий продукт

Висновок до розділу 8

Дана систма при своєму використанні дає можливість Підняти ефективність роботи групи станків з чпу. Можливість складати розклад та вести аналітику дозволить продукту легко скласти конкуренцію наявним на ринку аналогам.

В даний час на ринку представлено кілька схожих за своїми функціями та можливостями систем. Їх було детально проаналізовано, та знайдені їх сильні та слобкі сторони. Проаналізувавши отримані дані можна з впевненістю констатувати, що стартап являється перспективним. Аналіз технічної складової проекту виявив, що на ринку присутні в вільному доступі усі необхідні для його реалізації технології.

Було виявлено потенційний ринок збуту, та проведено його детальний аналіз. На основі аналізу ринку, можна винести вердикт, що ринок являється перспективним для того, щоб випустити на ньому стартап.

На основі аналізу успішності конкурентів, та їх користувачів, було визначено цільову аудиторію. Нею являються представники малого та середнього бізнесу.

Було розглянуто різноманітні загрози, що могли б виникнути на різних етапх життєвого циклу стартап проекту. Можна виділити дві основні групи загроз, це конкуренти, а точніше один з конкурентів, та постачальники програм управління станками з ЧПУ.

Маркетингова кампанія програмної системи забезпечить успішну реалізацію проекту в задані строки. В маркетинговій програмі продукту буде використано його основні переваги для кращої реалізації.

Каналом комунікації з користувачами було вирішено вибрати мережу Інтернет, та розгорнути в ній відповідні спеціальні ресурси.

ВИСНОВКИ

В роботі розроблено систему керування групою станків з ЧПУ (числовим програмним управлінням) на основі теорії розкладів та алгоритмів, які використовуються для розв’язання групи задач без переривань. Для створення системи використовувалась мова програмування C#, та принципи ООП (об’єктно-орієнтованого програмування).

В самій програмній частині системи було реалізована можливість роботи з зовнішніми бібліотеками, що відповідають за взаємодію основної програми та різних програм управління конкретними станками. Відповідно це дозволило зробити модульну структуру основної програми, залишивши в ній лише алгоритми створення розпорядку, а всю зовнішню взаємодію винести в окремі бібліотеки. Це в свою чергу полегшить оновлення програмної частини для роботи з новими станками з ЧПУ.

Для реалізації програмної частини системи, було використано стек технологій, що матиме в собі засоби програмування .NET Core, для реалізації використовувалась мова програмування C#. Програмний продукт, що було отримано має реалізацію інтерфейсу користувача на основі технології Windows Presentation Foundation (WPF), що дозволило створити графічний інтерфейс, а також виводити оператору інформацію в інтуїтивно зрозумілому вигляді (виводити не лише текстову інформацію про стан виробництва або станків ЧПУ, а й графічну інформацію в вигляді графіків, схем та діаграм).

Алгоритми теорії розкладів було застосовано для оптимізації розкладу роботи групи станків з ЧПУ. Було розроблено програмну частину для систему планування, та моніторингу роботи групи станків з числовим програмним управлінням. Тематика системи актуальна, адже кількість станків з ЧПУ в світі продовжує зростати, так само зростає кількість напрямків виробництва де їх використовують вже, або будуть використовувати в найближчий час. І програмна реалізація методів теорії розкладів, дозволяє підвищити ефективність їх роботи.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Система керування групою станків з ЧПУ. // Winter InfoCom 2020. – 2020.
2. Программы для станков с ЧПУ [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://www.multicut.ru/articles/programmy-dlya-stankov-s-chpu/.
3. Delcam ArtCAM Pro (RUS) x32-x64 bit [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://venemus.com/programms/1304-artcam.html.
4. LinuxCNC бесплатное ПО для ЧПУ станка [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: http://homecnc.ru/soft-cnc-stanok/20-linuxcnc.
5. Mach4 [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://www.machsupport.com/software/mach4/.
6. Программные продукты компании MR Soft [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://www.mariobad.ru/MrSoft.php.
7. L’empire Des Cnc [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://www.mariobad.ru/Cutviewer.php.
8. L’empire Des Cnc [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://www.mariobad.ru/Cutviewer.php.
9. ИНФОРМАЦИЯ [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://clobbi.com/ru/.
10. Анализ требований. [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: http://arm.ho.ua/531k/1analiztreb.html.
11. ITteach.ru [Електронний ресурс] / ITteach. – 2019. - Режим доступу до ресурсу: https://itteach.ru/bpwin/metodologiya-idef0.
12. .NET documentation [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://docs.microsoft.com/en-us/dotnet/fundamentals/.
13. What's new in .NET 5 [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://docs.microsoft.com/en-us/dotnet/core/dotnet-five.
14. Руководство по WPF [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://metanit.com/sharp/wpf/.

15. Руководство по Entity Framework [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://metanit.com/sharp/entityframework/.